6000 SCOPE 3 HANDOUTS

SECTION	TITLE
I.	INTRODUCTION TO 6000 SYSTEMS
II.	GENERAL 6000 SCOPE 3 CHARACTERISTICS
III.	DEAD START LOAD AND PRELOADER (PLR)
IV.	POOL PROCESSOR RESIDENT (PPR)
V.	MONITOR (MTR)
VI.	DYNAMIC SYSTEM DISPLAY (DSD)
VII.	STACK PROCESSOR (1SP/1SQ)
VIII.	JOB FLOW
IX.	SCOPE LOADER
X.	CIRCULAR INPUT/OUTPUT (CIO)
XI.	CENTRAL PROGRAM CONTROL (CPC)
XII.	CHECKPOINT/RESTART
XIII.	JOB DISPLAY
XIV.	SYSTEM AND JOB MAINTENANCE

INTRODUCTION

This document is intended primarily as an instructional handout and is intended to be used in the 6000 SCOPE 3.X Operating System course. It is not intended to replace the 6000 Scope 3.1 Reference Manual or the 6000 SCOPE 3.1 I.M.S. manual. Rather it is to be used with them.

NON-6000 SYSTEM PROGRAM FLOW

PROGRAMMER --- KEYPUNCH --- CARD READER → TAPE --- TAPE → CPU → TAPE --- TAPE → PRINTER --- PROGRAMMER

OFF-LINE (CARD READER → TAPE) | ON-LINE (TAPE → CPU → TAPE) | OFF-LINE (TAPE → PRINTER)

6000 SYSTEM PROGRAM FLOW

PROGRAMMER --- KEYPUNCH --- CARD READER → DISK → CPU → DISK → PRINTER --- PROGRAMMER

OFF-LINE (CARD READER → DISK) | ON-LINE (DISK → CPU → DISK) | OFF-LINE (DISK → PRINTER)

(DETAILS)

CARD READER → PPX → CM BUFFER AREA → PPX → DISK → PPX → CM PROGRAM AREA → CPU → CM PROGRAM AREA → PPX → DISK → PPX → CM BUFFER AREA → PPX → PRINTER

OFF-LINE (STEP I)

JOB NOW IN "INPUT JOB STACK"

ON-LINE (STEP II → STEP III)

JOB NOW IN "OUTPUT JOB STACK"

OFF-LINE (STEP IV)

PROGRAM FLOW CHART

PROGRAM FLOW STEPS

STEP I - LOAD JOB

A. CARD READER TO DISK OPERATION.
B. RECORD JOB NAME, PRIORITY, FL, AND WHERE PLACED ON DISK. IN CMR.
C. READ NEXT JOB (RETURN TO A)

STEP II - BEGIN JOB

A. SEARCH LIST OF JOBS PLACED ON DISK FOR THE HIGHEST PRIORITY JOB
B. IF FOUND, LOAD CONTROL CARDS TO CONTROL POINT AREA IN CMR.
C. TRANSLATE JOB CARD AND SET UP CONTROL POINT AREA.
D. SET UP CONDITIONS WHICH WILL INDICATE TO MTR TO INITIATE STEP III

STEP III - ADVANCE JOB

A. TRANSLATE NEXT CONTROL CARD AND CAUSE EXECUTION OF IT.
B. WHEN EXECUTION FINISHED, MTR CAUSES A RETURN TO "A".
C. LAST CARD EXECUTED, UNNEEDED FILES ARE DROPPED, EQUIPMENT IS DROPPED, JOB NAME AND PRIORITY ASSIGNED TO THE OUTPUT FILE, CLEAR OUT THE CONTROL POINT AREA.
D. CALL FOR STEP II's SYSTEM PROGRAM. (RETURN TO STEP II)

STEP IV - DUMP JOB

A. SEARCH FOR A JOB READY TO BE DUMPED
B. IF ONE FOUND, ASSIGN TO CONTROL POINTS "BUFFER POINT" AND DUMP THE JOB
C. DUMP THE DAYFILE FOR THE JOB ON THE PRINTER.
D. DROP THE FILE.

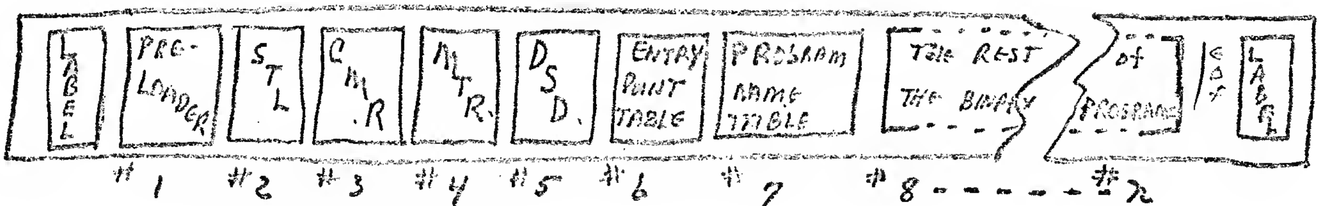
6000 SCOPE 3.1

I. 2 FILES MAKE UP THE OPERATING SYSTEM:

A. 1ST FILE IS THE BINARY DEAD START PORTION

1. 2 VERSIONS
   a. 6603 DISK FILE
   b. 6638 DISK FILE

2. FORMAT OF BINARY FILE -



(LOGICAL RECORDS)

B. 2ND FILE IS PROGRAM LIBRARY, "ØLDPL".
   1. CONSISTING OF;
      a. 1ST RECORD HAS;
         1) 1ST WORD CONTAINS 2 COUNTERS; COUNT OF THE IDENTIFIERS IN THE IDENTIFIER SECTION (BITS 35-18) AND THE COUNT OF THE DECK NAMES (17-00).

         2) A DIRECTORY WHICH CONTAINS A LIST OF ALL IDENTIFIERS USED. ("CORRECTION IDENTIFIERS")

         3) A DECK LIST WHICH CONTAINS A LIST OF ALL DECKS WHICH OCCURR OR HAVE OCCURRED ON THE FILE. ("DECK NAMES CURRENTLY KNOWN")

         4) THE TEXT STREAM WHICH CONTAIN THE CARD IMAGES AND CONTROL INFORMATION, KNOWN AS "CORRECTION HISTORY BYTES".

      b. THE BINARY DECKS OF THE PRODUCT SET, EACH CONSTITUTING A RECORD.

      n

      ENDING IN E.O.F.

PROGRAM LIBRARY CONTENTS

COMMON DECKS

COMFILE	the deck containing definitions of system symbols
FNTSRCH	a routine to search the FNT for a specified file name
READPP	a routine to read a record directly into PP memory
BUFINIT	a routine to initiallize PP memory for a call to READPP
RELOC	a set of macros which,if called,will cause a PP routine to be self-relocating
PPTIME	a routine to enter PP time into the system dayfile
COMCOMP	a micro which defines the COMPASS version number
TRNCOM	a set of declarations for use by the translator subroutines

TEXT DECKS

TAPEDS	program to generate the tape dead-start card
DISCDS	program to generate the disc dead-start card
DUMPDS	program to generate the dead-start dump cards
RECOV	program to generate the recovery dead-start card
PREPP	PP portion of the system pre-loader
PRECP	CP portion of the system pre-loader
STLPP	PP portion of the system dead-start loader
STLCP	CP portion of the system dead-start loader
CMR	central memory resident
MTR	system monitor
DSD	system display
CIO	central input/output package
LDR	relocatable loader
LOD	relocatable loader
MSG	routine to enter messages into the dayfile

RFL	routine to request field length
1AJ	routine to advance control point
1BJ	routine to bring a job to a control point
1LJ	routine to load a job from the card reader
1OT	routine to control printing and punching of output
1SP	6603 stack processor
2BP	routine to check buffer parameters (FET)
2EF	routine to process error flag
2ES	routine to enter a stack request
2RC	routine to read cards
2TR	routine to read 3000 magnetic tapes
2TS	routine to translate control card statements
2TW	routine to write 3000 magnetic tapes
ATS	STITCH routine
CHK	routine to check output file for RUN compiler
CKP	routine to take checkpoint dumps
CLL	STITCH routine
CLO	routine to close a file
CLS	STITCH routine
CPL	STITCH routine
CTS	routine to process COMMON function
DIS	job display
DMP	routine to dump central memory
EXU	STITCH routine

LBC	routine to load binary cards from INPUT
LOC	routine to load octal correctors from INPUT
MDI	routine to move system directory (EDITLIB)
MEM	routine to process MEMORY function
OPE	routine to open a file
PBC	routine to punch binary cards from central memory
RBR	routine to read a binary record
REQ	routine to process a REQUEST card or function
RST	routine to restart a job from a checkpoint file
SRB	routine to enter an expanded disc address into the directory (EDITLIB)
TIM	routine to process a TIME, DATE, CLOCK, or JDATE function
WBR	routine to write a binary record
1BT	routine to initiallize a blank tape
1CO	routine to complete OUTPUT file
1CY	routine to copy a file to the checkpoint file
1DF	routine to dump the dayfile
1LT	routine to load a job from magnetic tape
1MF	routine to open a tape file (multi-file tape)
1MR	routine to open a tape file (multi-reel tape)
1MW	routine to open a tape file
1OD	routine to open a file
1PL	routine to process plotter output (dummy)
1PO	routine to process punch output
1RI	routine to process the ROLLIN type-in

1RO	routine to process the ROLLOUT type-in
1SX	routine to enter stack processor error messages into the dayfile
1TD	routine to dump output files to tape
2CA	checkpoint abort procedure
2CF	routine to close files in preparation for the output queue
2CJ	routine to append the control point dayfile to the primary output file
2DF	routine to drop a file from the system
2LA	relocatable loader overlay
2LB	relocatable loader overlay
2LE	relocatable loader overlay to process error conditions
2LP	routine to process on-line print output
2PC	routine to process punch output
2RT	routine to read 6000 magnetic tapes
2TB	routine to handle backward motion on 3000 magnetic tapes
2TF	routine to handle forward motion on 3000 magnetic tapes
2TJ	routine to translate job cards
2WT	routine to write 6000 magnetic tapes
3OT	routine to process print output from the OUTPUT package
4LB	routine to handle label-processing
7TP	routine to handle write parity errors on 3000 magnetic tapes

LOADER	central memory portion of the relocatable loader
OVERLOD	abbreviated central memory loader for loading overlays
QXX	STITCH routine
EDITLIB	program for producing and editting system libraries
EDITSYM	program for producing and editting program libraries
COPYBCD	program for writing discrete print line images on magnetic tape
CPC	program for communicating requests from central memory to the system
IORANDM	routine for processing SCOPE style indexes
IO	routine to handle blocking/deblocking
COMPARE	routine to compare contents of two files
RESTART	routine to initiate the restarting of a job from a checkpoint file
BKSP	routine to backspace a file one or more logical records
COPY	routine to create a job stack on magnetic tape
COPYBF	routine to copy one or more binary files
COPYN	routine to copy selected records or files
COPYSBF	routine to format a packed display code binary file for printing
REWIND	routine to rewind a file
SYSTEXT	source deck of the system macros

DEFUNCT PP OVERLAYS

Name	Replacement
HLP	EDITLIB includes HLP's function
PBS	Punching in I-mode not necessary
SOS	EDITLIB includes SOS's function
1DS	Function included in DIS
1RF	Dead-start recovery card
2BD	Stack processor
2DT	Stack processor/central memory tables
2RD	Stack processor
2WD	Stack processor
3SD	Dayfile search is not necessary
4SD	Dayfile search is not necessary
7DP	Stack processor

COMFILE

The first of the common decks contained in the SCOPE program library is COMFILE. The purpose of COMFILE is to gather all system parameters together in one centralized location so that they may be easily examined and/or altered. Each parameter is referenced by a system symbol which consists of three parts:

-- an identifier of one or two characters denoting the category to which the symbol belongs;

-- "." a period following the identifier to indicate that this symbol is part of COMFILE;

-- a mnemonic of 1 to 6 characters suggesting the meaning of the symbol.

The definition of symbols of the above form should be avoided when COMFILE is to be called.

The system symbols in COMFILE reference the following categories of parameters:

-- installation parameters

-- system table lengths

-- system locations, words, and bytes

-- pointer words

-- PP resident entry points

-- monitor functions

-- quantities of system elements (number of tables, devices, etc.)

-- PP direct locations

The set of system symbols is easily expandable as new symbols are defined. The set is callable from each system routine by use of a single card:

*CALL,COMFILE

When COMFILE is called, a listing of COMFILE may be obtained by defining the symbol called LISTCOMF before the call to COMFILE.

SYSTEM SYMBOL IDENTIFIER DEFINITIONS

C. Most C.x symbols represent 12-bit byte positions within central memory words, where bytes are numbered from left to right as 0 through 4. C.x symbols are also used to represent first word addresses of PP overlays.

CP. CP.x symbols represent words or start of programs in the CP resident area.

D. The D.x symbols represent PP direct locations (low core); they are equated to 6-bit values from 00 through 77_8.

IP. All IP.x symbols represent installation parameters.

L. The L.x symbols represent table lengths or lengths of miscellaneous quantities.

LE. The LE.x symbols represent the lengths of entries within tables.

M. M.x symbols are equated to the values representing monitor functions.

F. F.x symbols are equated to error flag values.

CH. CH.x symbols represent pseudo-channel assignments; e.g., CH.FNT is the File Name Table channel.

N. N.x symbols represent quantities of things; e.g., N.DEVICE is equated to the number of allocatable devices within the system.

Ø. Ø.x symbols represent stack processor orders (commands). It should be noted that these orders do not correspond to values used in the code and status FET field: stack processor orders are designed for ease of use by the stack processor.

ØV. ØV.x symbols represent 3-character display code PP overlay names. There is one such symbol for each overlay; the mnemonic (x) is, in all cases, the 3-character overlay name. The appropriate ØV.x should be referenced whenever an overlay is to be loaded in order to force an entry in the cross reference table.

P. P.x symbols represent locations of central memory pointer words.

R. R.x symbols represent PP resident entry points.

S. S.x symbols represent the right offset of a field within a PP word, i.e. the number of bit positions which must be right shifted to right-justify the field to bit 0.

T. The T.x symbols are equated to the first word addresses of central memory tables. In general, these addresses should be obtained from the contents of the pointer words rather than direct use of T.x symbols.

W.x The W.x symbols are equated to values representing the relative positions of central memory words within tables. For example, assume that the address of a control point area is contained in the PP A-register; then to obtain the word containing the job name, the following code should be written

```
ADN  W.CPJNAM
CRD  D.TO
```

```
*              64/65/6600 SCOPE 3,0
********************************************************************
W,CPRES2  EQU    160B
*
*              SCOPE VERSION 3,0 COMMON SYMBOL DEFINITION
*              IN ORDER TO LIST THE CONTENTS OF OF COMFILE, THE
*              SYMBOL ≠LISTCOMF≠ SHOULD BE DEFINED PRIOR TO CALLING COMFILE
*
*
********************************************************************
               IF          =DEF,LISTCOMF,1
               LIST        =L
               EJECT
PCPADR    EQU  22000B
LCPADR    EQU  42000B
T,CPZ     EQU         20B
T,MON     EQU         21B
T,STO     EQU         22B
T,CIDLE   EQU         23B
T,PIDLE   EQU         24B
T,PPR     EQU         25B
T,CLK     EQU         30B
T,SLAB1   EQU         31B
T,DATE    EQU         31B
 T,JDATE  EQU          27B
T,SLAB2   EQU         32B
T,SLAB3   EQU         33B
T,SLAB4   EQU         34B
T,SLAB5   EQU         35B
T,SLAB6   EQU         36B
T,MSP     EQU         37B
 T,MSC    EQU          40B
T,CPS     EQU         40B
T,PPS0    EQU         41B
T,PPS1    EQU         42B
T,PPS2    EQU         43B
T,PPS3    EQU         44B
T,PPS4    EQU         45B
T,PPS5    EQU         46B
T,PPS6    EQU         47B
T,PPS7    EQU         50B
T,PPS8    EQU         51B
T,PPS9    EQU         52B
T,TMP     EQU         55B
T,CPT1    EQU         56B
```

```
LE,DFB3   EQU    100B                                                                        COMFILE00565
LE,DFB4   EQU    100B                                                                        COMFILE00566
LE,DFB5   EQU    100B                                                                        COMFILE00567
LE,DFB6   EQU    100B                                                                        COMFILE00568
LE,DFB7   EQU    40B                                                                         COMFILE00569
L,DFB     EQU  LE,DFB0+LE,DFB1+LE,DFB2+LE,DFB3+LE,DFB4+LE,DFB5+LE,DFB6+LECOMFILE00570
,,DFB7+100B                                                                                  COMFILE00571
T,LIB     EQU      T,DFB+L,DFB/10B*10B+10B                                         11/1      COMFILE00572
C,DIRPTR  EQU      0                                                                         COMFILE00573
C,DIRRBA  EQU      2                                                                         COMFILE00574
C,DIRRBN  EQU      3                                                                         COMFILE00575
C,DIRPRU  EQU      4                                                                         COMFILE00576
C,DIRCMA  EQU       1                                                              10/25     COMFILE00577
C,DIRUNT  EQU       1                                                              10/25     COMFILE00578
S,DIRPT   EQU       4                                                                        COMFILE00579
S,DIRPR   EQU      8                                                                         COMFILE00580
C,RBTWPL  EQU    0                                                                 11/16     COMFILE00581
C,RBTRBR  EQU    1                                                                 11/16     COMFILE00582
S,RBTRBR  EQU    3                                                                 11/16     COMFILE00583
C,RBTFB   EQU    1                                                                 11/16     COMFILE00584
S,RBTRND  EQU       6                                                                        COMFILE00585
S,RBTREL  EQU       7                                                                        COMFILE00586
C,RBTAL   EQU    2                                                                 11/16     COMFILE00587
C,RBTPRU  EQU    3                                                                 11/16     COMFILE00588
C,RBTLRB  EQU    4                                                                 11/16     COMFILE00589
C,RBTLPR  EQU    0                                                                 11/16     COMFILE00590
          EJECT                                                                              COMFILE00591
          LIST     L                                                                         COMFILE00592
          EJECT                                                                              COMFILE00593
```

SYSTEM LIBRARY + DEAD START LOAD PROCESS 2-1 + 2-2

The organization of the dead-start loader is as follows:

The first two records on the system tape are concerned with dead-starting. The first one is PLR (called preloader) and has the job of copying the tape to disc (or some allocatable device). The second is STL which actually loads the system from the allocatable device.

The programs are subdivided in the following way:

PLR The two major divisions are the portion that runs in the peripheral processors which is called PLRPP and the part that is executed by the central processor which is called PLRCP.

PLRPP

This is divided up into a tape-read program which WRITES INTO reads central memory, and a disc-write program which reads the information from central memory and writes it on the disc. At the end of the write program is a set-up program which is written on the disc at the end of preloading and is used to bootstrap in STL from disc.

PLRCP

This program accepts the information from the read program, checksums it, and arranges it in the proper format for the write program.

STL This record contains the PP resident and the recovery package as well as the programs to load. There are also two of these, STLPP and STLCP.

STLPP

This reads the information from disc and transfers it to central memory as well as sending MTR to PPO, DSD to PP9, and the resident to each of the other PP's.

STLCP

This receives the information from the STLPP program, checksums it, compares the checksum, and arranges all programs that are central memory resident in their proper place in central memory. It also inserts the addresses of all library programs into the directory.

Depending on the channel of the card reader, three different settings of the dead-start panel are necessary. One setting is used by people with card reader on channel 0 (if any such exist), another must be used for card reader on channels 1-11B, and the third for channels 12B or 13B.

The setting for channel 0 is:

Word on Panel	Setting	
0001	7700	
0002	E000	select card reader
0003	7700	
0004	1400	select input for EOR
0005	7400	activate channel
0006	7100	input on channel into location zero
0007	0000	

Words 0010 through 0014 are irrelevant.

Note: E is the equipment number of the card reader.

The setting for channels 12B or 13B is:

Word on Panel	Setting	
0001	75xx	
0002	77xx	
0003	E000	select card reader
0004	77xx	
0005	1400	select input to EOR
0006	74xx	activate channel
0007	71xx	input on channel into location zero
0010	0000	

Words 0011 through 0014 are irrelevant.

Note: xx = the card reader channel
E = the equipment number of the card reader

The setting for channel 1-11B is:

Word on Panel	Setting	
0001	1410	load A with 10B
0002	73xx	send part of panel out to another PP
0003	0006	
0004	75xx	disconnect channel
0005	7112	leave this PP on channel 12
0006	0000	
0007	77xx	
0010	E000	select card reader
0011	77xx	
0012	1400	select input to EOR
0013	74xx	activate channel
0014	71xx	input into location zero

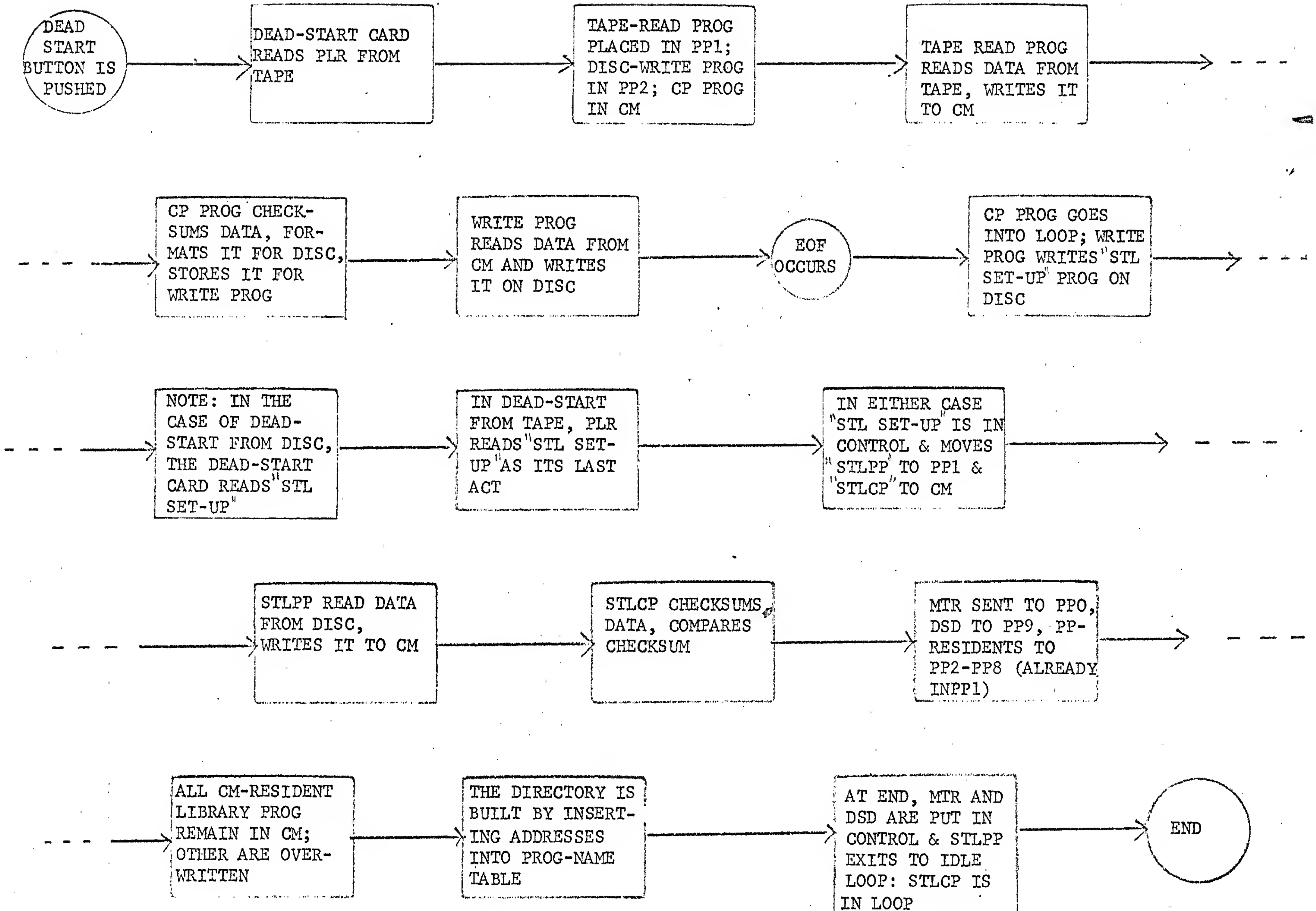
Note: The dead-start process adds a word of zeros at the end of the panel so this is why the last instruction will read into location zero.

A [illegible] setting for dead-starting directly from tape without the use of cards is also provided.

Word on Panel	Setting	
0001	75xx	disconnect channel
0002	77xx	
0003	E00u	select tape unit
0004	77xx	
0005	0010	rewind tape
0006	77xx	
0007	1400	select input to EOR
0010	74xx	activate channel
0011	71xx	
0012	0073	read record into location 73B
0013	0112	jump to beginning of record + 7
0014	0007	

Note: This can only be used is the tapes are on channels 12B or 13B.

xx = the channel of the tape
E = the equipment number of the tape
u = the unit of the tape


DEAD START BUTTON IS PUSHED
DEAD-START CARD READS PLR FROM TAPE
TAPE-READ PROG PLACED IN PP1; DISC-WRITE PROG IN PP2; CP PROG IN CM
TAPE READ PROG READS DATA FROM TAPE, WRITES IT TO CM
CP PROG CHECK-SUMS DATA, FOR-MATS IT FOR DISC, STORES IT FOR WRITE PROG
WRITE PROG READS DATA FROM CM AND WRITES IT ON DISC
EOF OCCURS
CP PROG GOES INTO LOOP; WRITE PROG WRITES "STL SET-UP" PROG ON DISC
NOTE: IN THE CASE OF DEAD-START FROM DISC, THE DEAD-START CARD READS "STL SET-UP"
IN DEAD-START FROM TAPE, PLR READS "STL SET-UP" AS ITS LAST ACT
IN EITHER CASE "STL SET-UP" IS IN CONTROL & MOVES "STLPP" TO PP1 & "STLCP" TO CM
STLPP READ DATA FROM DISC, WRITES IT TO CM
STLCP CHECKSUMS DATA, COMPARES CHECKSUM
MTR SENT TO PPO, DSD TO PP9, PP-RESIDENTS TO PP2-PP8 (ALREADY INPP1)
ALL CM-RESIDENT LIBRARY PROG REMAIN IN CM; OTHER ARE OVER-WRITTEN
THE DIRECTORY IS BUILT BY INSERT-ING ADDRESSES INTO PROG-NAME TABLE
AT END, MTR AND DSD ARE PUT IN CONTROL & STLPP EXITS TO IDLE LOOP: STLCP IS IN LOOP
END

CENTRAL MEMORY LAYOUT

SUMMARY OF CM RESIDENT AREAS

The Central Memory Resident Pointer Area contains pointers to larger tables contained elsewhere in CM, small tables, and various flags.

The PP Communication Area contains ten 8-word areas, one for each PP, through which PP's communicate with each other and Monitor. The Monitor communication area (T.PPC10) is not used.

The Control Point Area contains seven 200_8-word areas, one for each control point. This area is used to contain the exchange package, job name, and information about the job which is running at that control point.

The CP Resident area contains two programs which run at control point zero (the storage move program, and the idle package); their exchange packages are also kept in the CP Resident area.

The Equipment Status Table contains one entry for each device (allocatable or non-allocatable) attached to the system. Non-allocatable devices are those which may be assigned to a single control point, e.g. magnetic tape unit; allocatable devices may be used by many control points simultaneously.

The File Name Table is composed of as many 3-word entries as there are files in the system. An FNT entry is set up at the time that a file is created; it is not accessible to the user. This table provides a linkage between the user program and all required I/O tables and functions.

The Record Block Reservation area contains at least one Record Block Reservation table (RBR) for each allocatable device attached to the system. Each table contains a series of bits which denote whether or not the record block which it represents is assigned.

The Request Stack area is actually composed of two tables: The Device Status Table (DST) and the request stack itself. The DST contains one 2-word entry for each allocatable device within the system; it is static and its contents are defined at assembly time. The request stack contains entries which are requests for data transfers, device positioning, or logical operations on a file. Each entry is two words in length. The table grows from high memory to low.

The Catalogue and Security Password Index are currently of zero length; they are reserved for future use.

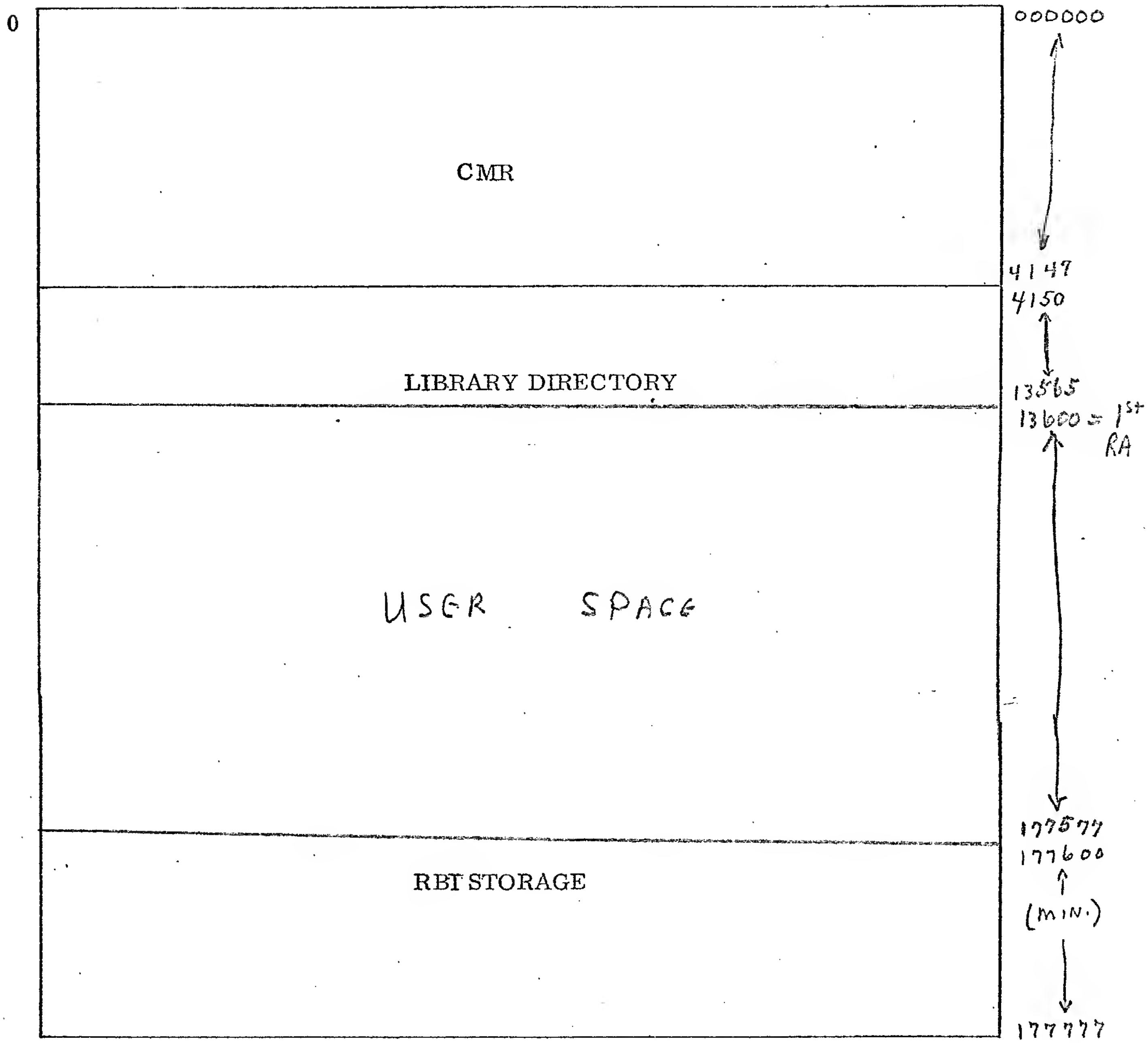
The Installation Area is currently of zero length; it is reserved for installation use.

The Dayfile Buffer area contains eight File Environment Tables and eight buffers, one for the system dayfile and one for each of the seven control points.

The library directory is composed of three sections: The entry point table, containing 1-word entries; the program name table, containing 2-word entries; and the bodies of the CM resident programs. The directory may be expanded or contracted as programs are added or deleted or as program residence is changed.

The Record Block Table area (RBT) is a collection of individual file chains, one for each file on an allocatable device currently recognized by the system. When a file is initiated, a single two-word RBT entry is assigned to that file; additional entries are assigned as needed. Each entry is divided into ten 12-bit bytes, some of which are used as pointers to additional entries, other tables, etc. The remaining bytes each contain the number of a particular record block assigned to the file in the physical order of their assignment. A record block number in an RBT is the ordinal of the bit in an RBR which represents that record block. The RBT empty chain is a pool from which words may be extracted to construct file chains and to which words are returned when the chains are discarded. The RBT expands and contracts by 100_8-word blocks as files are created and released.

Overview Central Memory Allocation (65K)



CMR

Address	Area
0	POINTERS
60	PP COMMUNICATION AREA
200	CONTROL POINT AREA
2000	CP RESIDENT
2040	EQUIPMENT STATUS TABLE (EST)
2140	FILE NAME/STATUS TABLE (FNT/FST)
3050	RECORD BLOCK RESERVATION TABLE (RBR)
3120	REQUEST STACK
3430	CATALOGUE*
3430	SECURITY PASSWORD INDEX*
3430	INSTALLATION AREA (INS)
3430	DAYFILE BUFFER (DFB)
4440	

* ZERO LENGTH — TO BE USED IN FUTURE DEVELOPMENT

CENTRAL MEMORY RESIDENT

P.ZERO	ZEROS					0
P.LIB	C.DIRFWA FWA of directory		LWA+1 of directory		dead-start load flag	1
P.RBR/P.RBT	C.RBRAD FWA of RBRs		FWA/2 of empty chain	length/100B of RBT	(LWA+1)/100B of memory	2
P.DFB	FWA/10 of dayfile					3
P.FNT	FWA of FNT	LWA+1 of FNT				4
P.EST	FWA of EST	LWA+1 of EST				5
						6
P.INS	installation use					7
						10
P.CAT	future development					11
P.SP1	future development					12
P.RQS	C.RQSCS stack entry word pair ct.		C.RQSFS FWA/2 of request stack	number of devices	FWA/10B of DSTs	13
T.CST	Channel 0	Channel 4	Channel 10	FST Channel		14
P.CST2	Channel 1	Channel 5	Channel 11	FNT Channel		15
P.CST3	Channel 2	Channel 6	Channel 12	LIB CHANNEL		16
P.CST4	Channel 3	Channel 7	Channel 13	RBT Channel		17
T.CPZ	0003		Storage Move Flag		Machine Field Length	20
T.MON	* MONITOR * 55					21
T.STO						22
T.CIDLE			seconds		milliseconds	23
T.PIDLE			seconds		milliseconds	24
T.PPR						25
						26
T.JDATE	0000	0000	00 Y Y D D D			27

Label	Contents					Word
T.CLK	* H H . M M . S S *					30
T.DATE	b M M / D D / Y Y b					31
T.SLAB2 ⋮ T.SLAB6	SCOPE VERSION 3.X					32 ⋮ 36
T.MSP					Step Flag	37
T.MSC	future development	seconds	milliseconds mod 1000_{10}	millisecond clock		40
T.PPS0 ⋮ T.PPS9			(MILLISEC. SINCE LAST SECOND UPDATE)	seconds	milliseconds	41 ⋮ 52
						53
T.CPZECS						54
T.TMP	MTR Scratch					55
T.CPT1/T.UAS	Unassigned CM/100B	Unassigned ECS/1000B	NNCX	CPU-B ACTIVITY	CPU-A Activity	56
						57

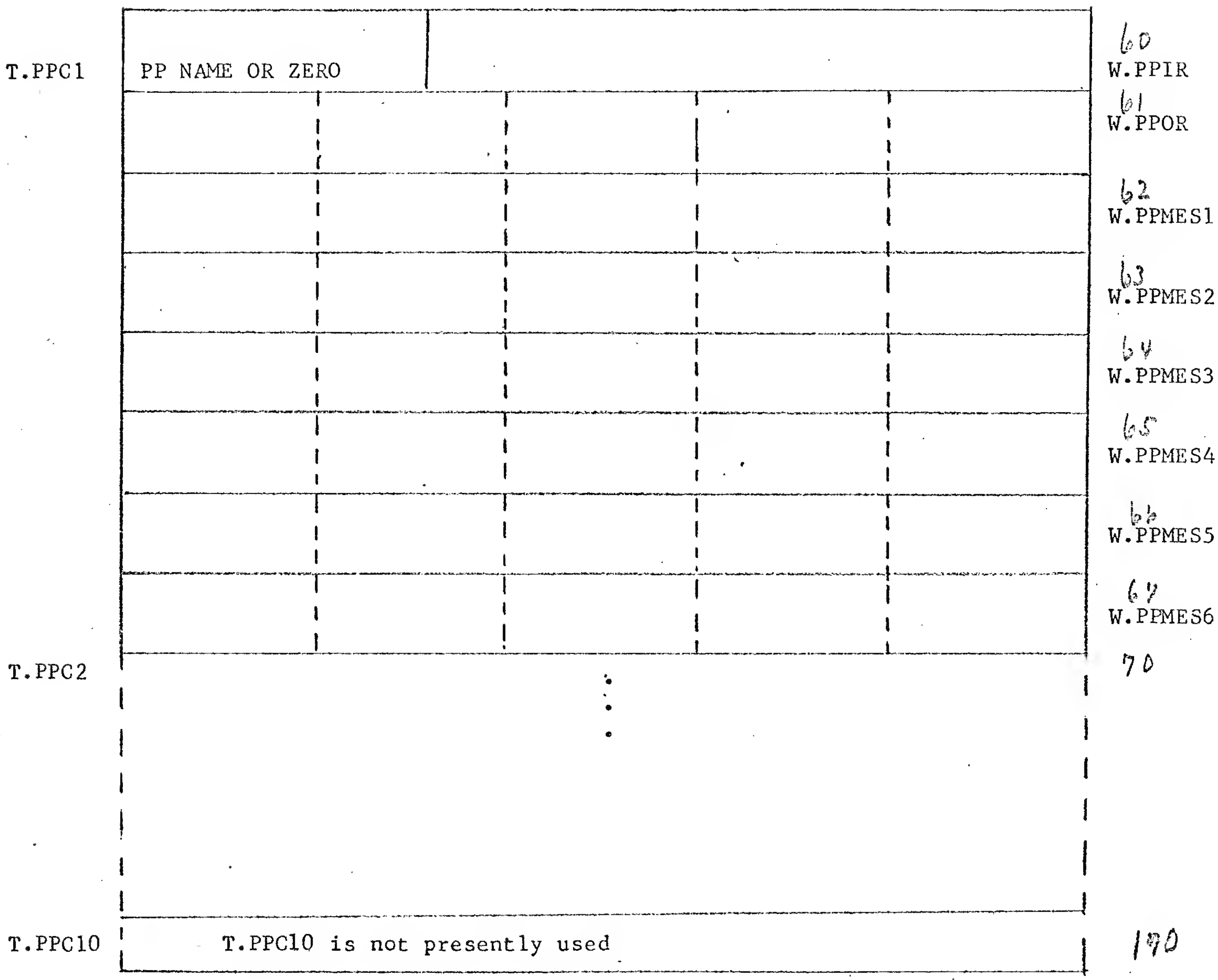
NN = CP #

C = 1 ⇒ CPU OFF
2 ⇒ DEDICATED
3 ⇒ DOESN'T EXIST

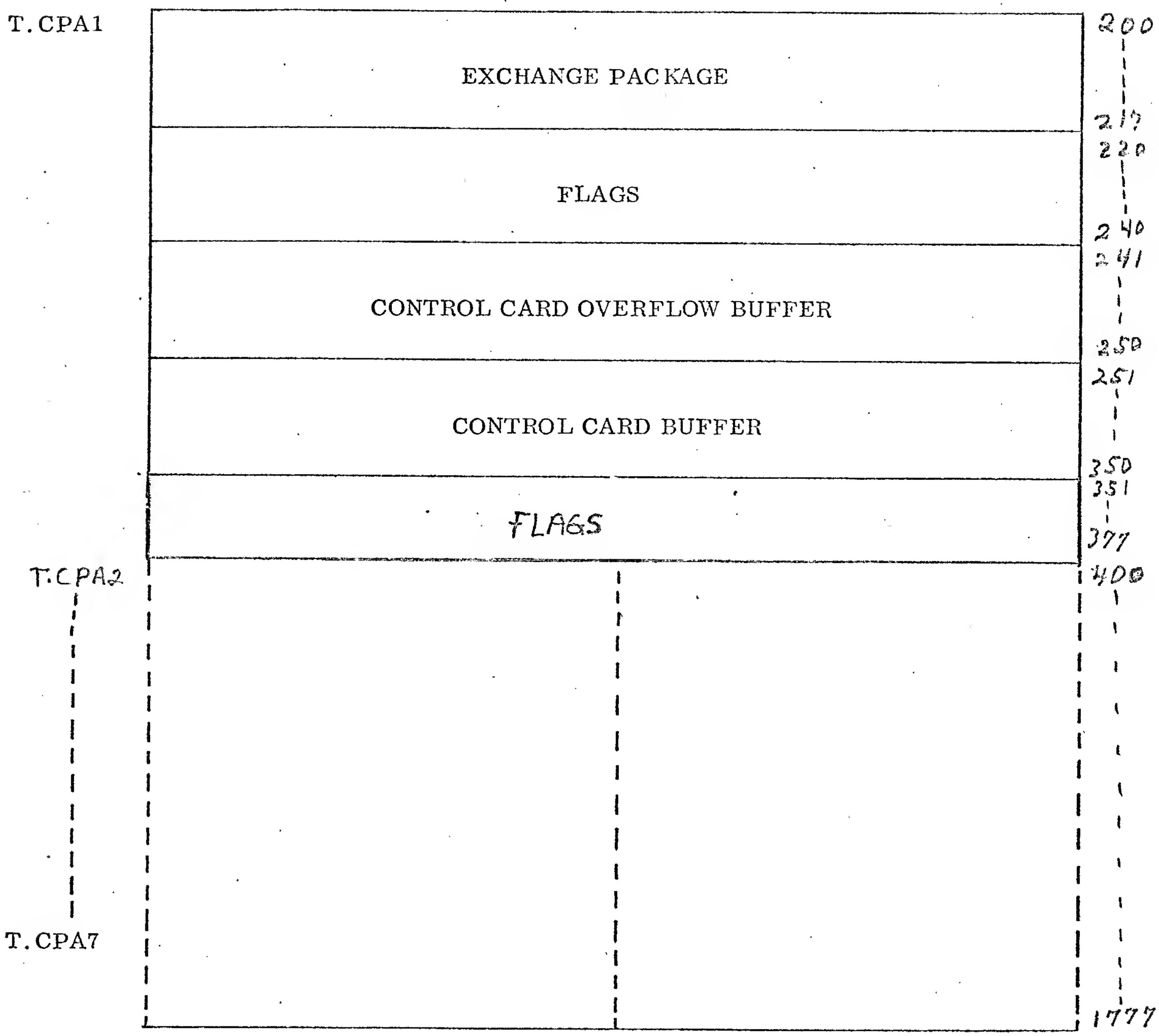
X = 1 ⇒ CPU-A
2 ⇒ CPU-B

2000 ⇒ STORAGE MOVE
0000 ⇒ IDLE PROGRAM
0200 ⇒ ASSIGNED TO CP1
0400 ⇒ ASSIGNED TO CP2
⋮
1600 ⇒ ASSIGNED TO CP7

PERIPHERAL PROCESSOR COMMUNICATION AREAS



CONTROL POINT AREAS



EXCHANGE PACKAGE

				Words
	Program Address(P)	A0(Address Registers)	0 0 0 0 0 0	0
	Reference Address(RA)	A1	B1(Increment Register)	1
	Field Length(FL)	A2	B2	2
	Exit Mode(EM)	A3	B3	3
RA - ECS		A4	B4	4
FL - ECS		A5	B5	5
	M A	A6	B6	6
		A7	B7	7
X0 (Operand Registers)				10
X1				11
X2				12
X3				13
X4				14
X5				15
X6				16
X7				17

CONTROL POINT AREA

Word name	59 – 48	47 – 36	35 – 24	23 – 12	11 – 00	Address
W.CPSTAT/ W.CPEF	C.CPSTAT (Status)	C.CPEF (Error Flag)	C.CPSM (Storage Move)	C.CPRA (RA/100B)	C.CPFL (FL/100B)	20
W.CPJNAM	JOB NAME				Next Control Statement	21
W.CPECS/W.CPPRI/ W.CPECS	Priority		Time Limit	ECS RA/1000B	ECS FL/1000B	22
W.CPTIME	millisecond count since last priority RE-EVALUATION		C.CPTIML CPU Time (seconds)		milliseconds	23
W.PPTIME		Loader Flag	PPU Time (seconds)		milliseconds	24
W.CPRCL	PP Recall (Obsolescent)					25
W.SSW	Sense Switches/Lights					26
W.EQP	Equipment Assignments					27
W.CPDFM	Last Dayfile Message					
W.CPERT			C.CPFST (FST Address)	C.CPFP	C.CPERT	40
W.CPTBUF	Control Card Overflow Buffer					41
W.CPTBUL						50
W.CPCAF	Control Card Buffer					51
W.CPCAL						150
W.FSTCC	FST Entry for Next Control Card PRU					151
W.FSTUR						152
W.CPOAE					Op. Assigned Equipment	153
W.CPVRNO	Visual Reel Number					154
W.CPENC	Stack Entry/ Exit Count	PP Job Entry Count				155
W.CPRES1	(USED BY RESPONSE)					156
W.CPAR	Auto-Recall Pointer					157
W.RES2	(USED BY RESPONSE)					160
						161
						:
						177

FST ADDRESS FOR CHECKING STATUS CODE

CONCORDANCE OF CONTROL POINT FIELDS, 2.0/3.0

Word	Mnemonic	Changed	Comments
20	W.CPSTAT, etc.	no	
21	W.CPJNAM	no	
22	W.CPPRI, etc.	yes	Msg. count removed, Time Limit moved left one byte Operator assigned equipment move, RA/1000B of ECS in byte 3, FL/1000B of ECS in byte 4
23	W.CPTIME	yes	bytes 0 and 1 contain a msec count since last priority re-evaluation (altered by MTR only)
24	W.PPTIME	yes	express flag removed
25	W.CPRCL	no	this work is being phased out in favor of MTR function 37 (see IMS)
26	W.SSW	no	
27	W.EQP	no	
30-37	W.CPDFM	no	
40	W.CPERT	yes	now contains pointers and flags for control card processing in bytes 2, 3, 4
41-50	W.CPTBUF	no	Control Card overflow buffer
51-150	W.CPTBUL	no	Control Card buffer
151	W.FSTCC	yes	FST entry for next control card PRU
152	unused	yes	
153	W.CPOAE	yes	Operator assigned equipment (see 22) in byte 4
154	W.CPVRN ~~unused~~	yes	
155	W.CPENC	yes	I/O stack entry count in byte 0, PP Job Buffer Entry Count in byte 1
156	unused	yes	

CONCORDANCE OF CONTROL POINT [illegible]

Word	Mnemonic	Cleared	Comments
157	W.CPAR	yes	Auto Recall pointer in bytes 0-1
160-177	unused	yes	

3-15-1

```
CP.SM   SB7   1
        NG    B3,DOWN
        SB6   -2
        SA1   B2-B7
        SA2   B2+B6
        EQ    LOOP
DOWN    SB6   2
        SA1   B1
        SA2   B1+B7
LOOP    SA4   A1+B6
        SA5   A2+B6
        BX6   X1
        LX7   X2
        SA6   A1+B3
        SA7   A2+B3
        SB1   B1+4
        SA1   A4+B6
        SA2   A5+B6
        BX6   X4
        LX7   X5
        SA6   A4+B3
        SA7   A5+B3
        LT    B1,B2,LOOP
        JP    0
```

STORAGE MOVE PROGRAM

```
         IFNE      IP.MECS,0
*                  MOVE EXTENDED CORE STORAGE
*
*                  PARAMETERS PASSED IN EXCHANGE PACKAGE
*
*        P         CP.ECSM
*        RA(CM)    0
*        FL(CM     400000B
*        EM        0
*        RA(ECS)   0
*        FL(ECS)   10000000B
*
*        B1        RA(ECS) OF CTL PT TO BE MOVED /100B
*        B2        RA+FL (ECS) /100B
*        B3        DISPLACEMENT /100B
*        B4        LENGTH OF CM BUFFER AREA
*        A0        ADDRESS OF CM BUFFER AREA
*
CP.ECSM  SX1       B1                INITIALIZE THE X REGISTERS
         LX1       6
         SX2       B2
         LX2       6
         SX3       B3
         LX3       6
         SX5       B4
         LX5       6
         NG        X3,SM34           IF DISPLACEMENT IS NEG, SHUTTLE DOWN
         EQ        SM24         ELSE, SHUTTLE UP
         SPACE     3
*                  SHUTTLE UP LOOP  (ECS)
SM18     IX6       X5-X4             WHEN THE REMAINING PORTION TO BE
         NG        X6,SM20           MOVED IS LESS THAN THE BUFFER
         BX5       X4                REDUCE THE BUFFER SIZE
SM20     SB5       X5
         IX0       X2-X5             ECS ADDRESS OF THE MOVE
         BX2       X0
         RE        B5+0              READ INTO THE CM BUFFER
         JP        *
         IX0       X0+X3             ADJUST THE ECS ADDRESS
         WE        B5+0              WRITE BACK INTO ECS
         JP        *
SM24     IX4       X2-X1             CONTINUE TO LOOP UNTIL THE ENTIRE
         NZ        X4,SM18           BLOCK HAS BEEN MOVED,
         JP        0                 THEN EXIT
```

3-15-2

```
         SPACE     2                                          CMR   00381
*                  SHUTTLE DOWN LOOP  (ECS)                   CMR   00382
SM26     IX6       X5-X4                                      CMR   00383
         NG        X6,SM28                                    CMR   00384
         BX5       X4                                         CMR   00385
SM28     SB5       X5                                         CMR   00386
         BX0       X1                                         CMR   00387
         IX1       X1+X5                                      CMR   00388
         RE       B5+0                                        CMR   00389
         JP        *                                          CMR   00390
         IX0       X0+X3                                      CMR   00391
         WE       B5+0                                        CMR   00392
         JP        *                                          CMR   00393
SM34     IX4       X2-X1                                      CMR   00394
         NZ        X4,SM26                                    CMR   00395
         JP        0                                          CMR   00396
         ENDIF                                                CMR   00397


         BSSZ      T.EST-*                                    CMR   00399
```

3-15-3

EQUIPMENT STATUS TABLE

POINTER

P.EST	T.EST	T.EST + L.EST	0 ——— 0	5
		100_8		

EST

	59–48	47–36		35–24			23–12	11–0		
T.EST										
3000 Type Nonallocatable	Z	bb	aa	dd	cc	o	h	s	e	u
6000 Type Nonallocatable	Z		cc	e Ouu		o	h	0000		
Allocatable	4000 Z	O		O		o	h	O		

59 47 35 23 11 0

2040 => 100

L.EST WORDS IN LENGTH

Z - Status of Entry

aa, bb, cc, dd - Channel Numbers

o - ON/OFF Bit

h - hardware type

s - 6681 number

e - equipment

u - unit

4000 => allocatable Device

6000 => "non allocatable, and unassigned" or "no entry"

0200
0400
0600
1000
1200
1400
1600
} nonallocatable and assigned.

(3.2)

6000 + 3000 Allocatable.

FILE NAME TABLE ENTRY

WORD "1" (FNT)

FILE NAME	LOCK	TYPE		CONTROL POINT	PRIORITY
59–18	17	16	15	14–12	11–00

WORD "2"

READ CARD FILE

EQUIP MENT CODE	RECORD COUNT	RECORD TYPE	EST ORDINAL	NOT USED	EOJ	CARD COUNT WITHIN RECORD 1st = 001
59–54	53–(48)	(47)–36	35–24	23–18	17	16–00

INPUT FILE

EQUIP MENT CODE	NOT USED	FIRST RBT WORD PAIR	ECS FIELD ENTRY/1000	CURRENT RBT ENTRY	CURRENT RBT BYTE	CM FIELD LENGTH/100
59–54	53–48	47–	35–24	23–	–12	11–00

DISC FILES (RANDOM ALLOCATABLE FILE)

EQUIP-MENT CODE	NOT USED	CK	FIRST RBT WORD PAIR	CURRENT RBT WORD PAIR	CURRENT RBT ENTRY	CURRENT	RBT BYTE	CURRENT PRU NO.
59–54	53–49	48	–36	35–24	23–	15–14		11–00

TAPE FILES INDIVIDUAL SEQUENTIAL FILE

EQUIPMENT CODE		X	LABEL	DENSITY	EST ORDINAL OF SECONDARY DEVICE	EST ORDINAL OF PRIMARY DEVICE	CURRENT PRU NO.
59–54	53–52	51	50	49–48	47–36	35–24	23–00

MULTI-FILE FNT/FST ENTRY

EQUIP. CODE		LABEL	TAPE	DENSITY	SECONDARY DEVICE EST ORDINAL	PRIMARY DEVICE EST ORDINAL	POSITION # OF THE LAST FILE WRITTEN IN THE SEQUENCE OF FILES.	CURRENT FILE POSITION
59–54				–48	47–36	35–24	23–12	11–00

PUNCH CARD FILE

EQUIP MENT CODE		EST ORDINAL	FLAG USED BY 2PC	EOJ	CARD COUNT
59–54	53–36	35–24	23–18	17	16–00

WORD "3" (FST 2)

0	FUTURE USE	~~LAST USED~~ FET ADDRESS	DISPOSITION CODE	SE-CURITY CODE	LAST CODE AND STATUS
59	58–54	53–36	35–24	23–18	17–00

NOTE: On multi-reel files, the FNT word "3", bits 18+19 are always set (File always "CLOSED"), bits 24-35 = 0002 (multi file disposition), and bits 48 and 49 are used for logical security code.

Record type continued.

002 – 12.0₈ ⇒ "Free-form" card record
(See Ref. Man. Appendix E)
177 ⇒ Special mode ended.

Record Count within this file.

Equipment Code. See table on Page 3-4 and 3-5 of Ref. Manual. (60_8 ⇒ card reader, etc.)

CM Field Length/100_8 for this file.

ECS Field Length/1000_8 for this file.

Current RBT entry. This is a count of the number of full RBT word pairs up to the current position of the file. For example, if the file is positioned in the second RBT word pair, the current RBT entry will be 1. Just as soon as the second RBT pair fills up, this number will roll over to 2.

Current RBT Byte indicates which byte in the RBT word pair contains the RB address.

Current RBT word pair position. If this number = N, the address of the first word of this word pair is 400,000B-2*N.

First RBT word pair associated with this file. If this number = M, the address of the first word of this word pair is 400,000B-2*M.

Check point flag. is used with Checkpoint/Restart files.

Current PRU # indicates which sector is currently being used.

DEVICE STATUS TABLE ENTRY

WORD 1

NAME OF PP PROGRAM (59–42)		ENTRY COUNT (35–24)	23–18	17–12	11–6	5–0

SECONDARY CHANNEL # (bits 23–18)

PRIMARY CHANNEL # (bits 17–12)

DST ORDINAL (bits 5–0)

WORD 2

HEAD POSITION #1	HEAD POSITION #2	EXIT COUNT	23–18	17–12		6–0

SECONDARY CONTROLLER # (bits 23–18)

PRIMARY CONTROLLER # (bits 17–12)

PP OUTPUT REGISTER ADDRESS (bits 6–0)

REQUEST STACK ENTRY

WORD 1

FNT ADDRESS	39	ORDER		
RBT WORD, BYTE PAIR		(SEE SECTION 7)		

CONTROL POINT

DST ORDINAL

WORD 2 (FOR CM TRANSACTIONS)

FET ADDRESS	FWA		LWA+1

23	22		20	19	18

AVAILABLE BIT FOR 150/350 (bit 18)

FET BIT (bits 20–19)

FNT BIT

RELEASE/EOF BIT (bit 22)

RECALL BIT (bit 23)

WORD 2 (FOR PP TRANSACTIONS)

59–48	47–36	35 PP FWA 24	23–18	17–12	11 PP LWA+1 0

PP MESSAGE BUFFER ADDRESS (bits 47–36)

PP WORD COUNT (bits 59–48)

Record Block Reservation Tables

A Record Block Reservation (RBR) table serves a dual purpose:

1. It defines record blocks in a specific allocatable area.
2. It identifies those record blocks within that area as available or not available.

A maximum of 2048 record blocks may be defined by one RBR. The area described by an RBR must all be on one device.

The RBR tables in the SCOPE Operating System follow immediately after the FNT/FST area. The pointer to the first word of the first RBR table is found in Word 2, Byte 1 (counting 0-4, left to right) of low core. The second RBR table starts immediately after the first. The total length of each table is 38_{10} CM words. Thus, adding 46 (octal) to the pointer in Word 2, Byte 1 will give the starting address of the second RBR table.

The header words (first two) of each RBR table are comprised of the following fields:

Word

Bits 0 -11: Allocation type link. The ordinal of an RBR which references a device of the same type and which uses the same record block definition. The linkage should be circular.

Bits 12-23: Device link. The ordinal of an RBR which references a device of the same type. The linkage should be circular so that all RBR's referencing the same device type are linked.

RECORD BLOCK RESERVATION TABLE (RBR)

T.RBR

(RBR HEADER WORDS)

59	47	41	36	23	17	11	5 0
DEVICE TYPE (EQUIP. CODE)	DST ORDINAL	UNIT NUMBER	STARTING ADDRESS OF RB GROUP (FOR NON-ALLOC. DEVICES ONLY OTHERS = 0)	(ALLOCATION TYPES AVAILABLE IN THIS RBR) a	b	c *	d

3240

59 48	47 41	42 36	35 24	23 12	11 6	5 0
TOTAL NUMBER OF RECORD BLOCKS		EST ORDINAL	NUMBER OF PHYSICALLY AVAILABLE RECORD BLOCKS	NUMBER OF LOGICALLY AVAILABLE RECORD BLOCKS	CONT. #	CHAN. #

3½ WORDS

59 — WORD 0 — 00

32 WORDS

59 — WORD 37/8 — 00

4 TAIL WORDS

0		1		2		3		4
00	01	02	03	04	05	06	07	
10	11	12	13	14	15	16	17	
20	21	22	23	24	25	26	27	
30	31	32	33	34	35	36	37	

BYTE (BOOT) — 0, 1, 2, 3, 4

LEFT BOOT

RIGHT BOOT

64 BIT WORD

EX. 2^{59} — WORD 14 — 2^{00} | 2^{35}–2^{32}

TAIL WORD 1, BOOT 2

SEE PAGE 3-21-4 for order of reservations.

*. If standard allocation for the device is included, it is listed first, and the last 6-bit field is left zero. If all allocations are non-standard, all 4 fields must be non-zero. (Repeat last allocation type to end).

The allocation type link and device link may be broken up into 2 bytes of the form: ☆⇒ (see SYSTEMS BULLETIN #10, Pages 25-27)

aabbccdd=ALLOC

where each term ii specifies a legitimate allocation within the portion of the device indexed by this RBR.

01=50 PRU's/RB.

02=64 PRU's/RB.

00=Free allocation; when the user does not specify allocation in an OPEN call, the file may be assigned to any RBR with some ii=00. In this case, allocation aa is assigned to the file after the RBR is chosen, thus, aa should not be 00.

03=Free allocation; this allocation style is used to permit continuation of a file from one device to another (theoretically).

10=8 PRU's/RB (Used by RESPOND).

aa determines the actual RBR type for physical allocation purposes. ALLOC need not be supplied. Default values are:

for TYPE=1 (bits 48-50)

ALLOC=03020100

for TYPE=2

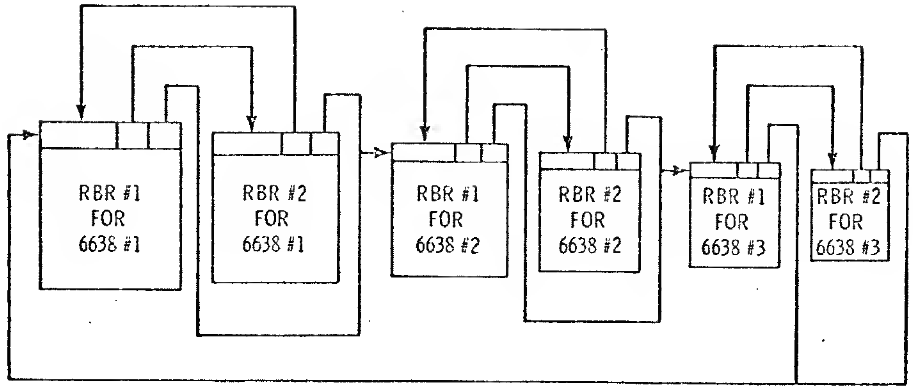
ALLOC=03010100

Bits 24-35: RB group start address - the half-track on which the described area starts.

Bits 36-41: Unit number.

Bits 42-47: Device number. The ordinal in the device status table (DST) of the device to which this RBR refers.

Bits 48-59: Device group. The device type and allocation type used in the area described by this RBR.



6638 RBR LINKAGES FOR 3 6638's

Word TWO

Bits 0 - 5: Channel number. The channel used by the device referenced by this RBR.

Bits 6 -11: Controller number.

Bits 12-23: Logical availability. The number of record blocks not assigned to files.

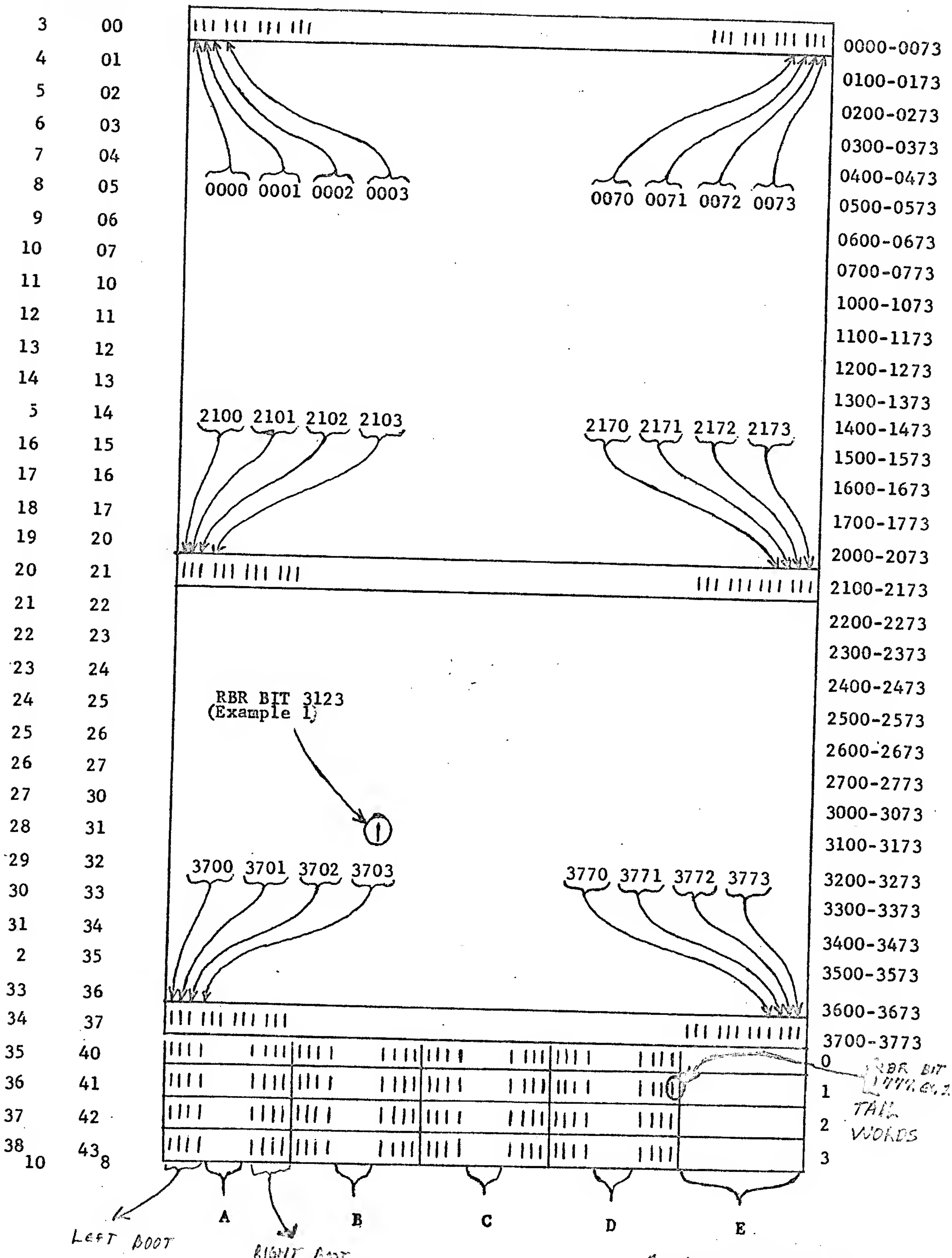
Bits 24-35: Physical availability. The number of usable record blocks. This is normally the same as the total RB count.

Bits 36-41: EST ordinal.

Bits 42-47: Not used.

Bits 48-59: Total RB count. The number of record blocks defined by this RBR.

RBR WORDS 0-37₈ 3-21-4

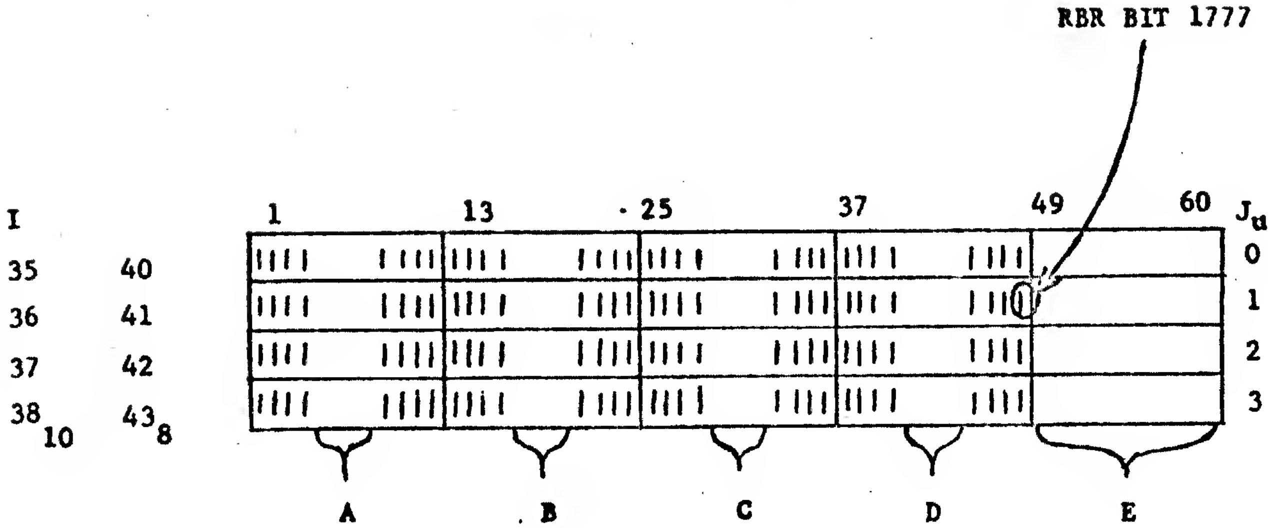


A—E ARE NOT USED.

3-21-5

RBR "TAIL WORDS"

(The last 4 words in an RBR Table.)



The bits designated by A, B, C, D, and E are not used.

A = Bits 52 - 55 (as usually numbered) or 5 - 8 (as numbered above)

B = Bits 40 - 43 (as usually numbered) or 17 - 20 (as numbered above)

C = Bits 28 - 31 (as usually numbered) or 29 - 32 (as numbered above)

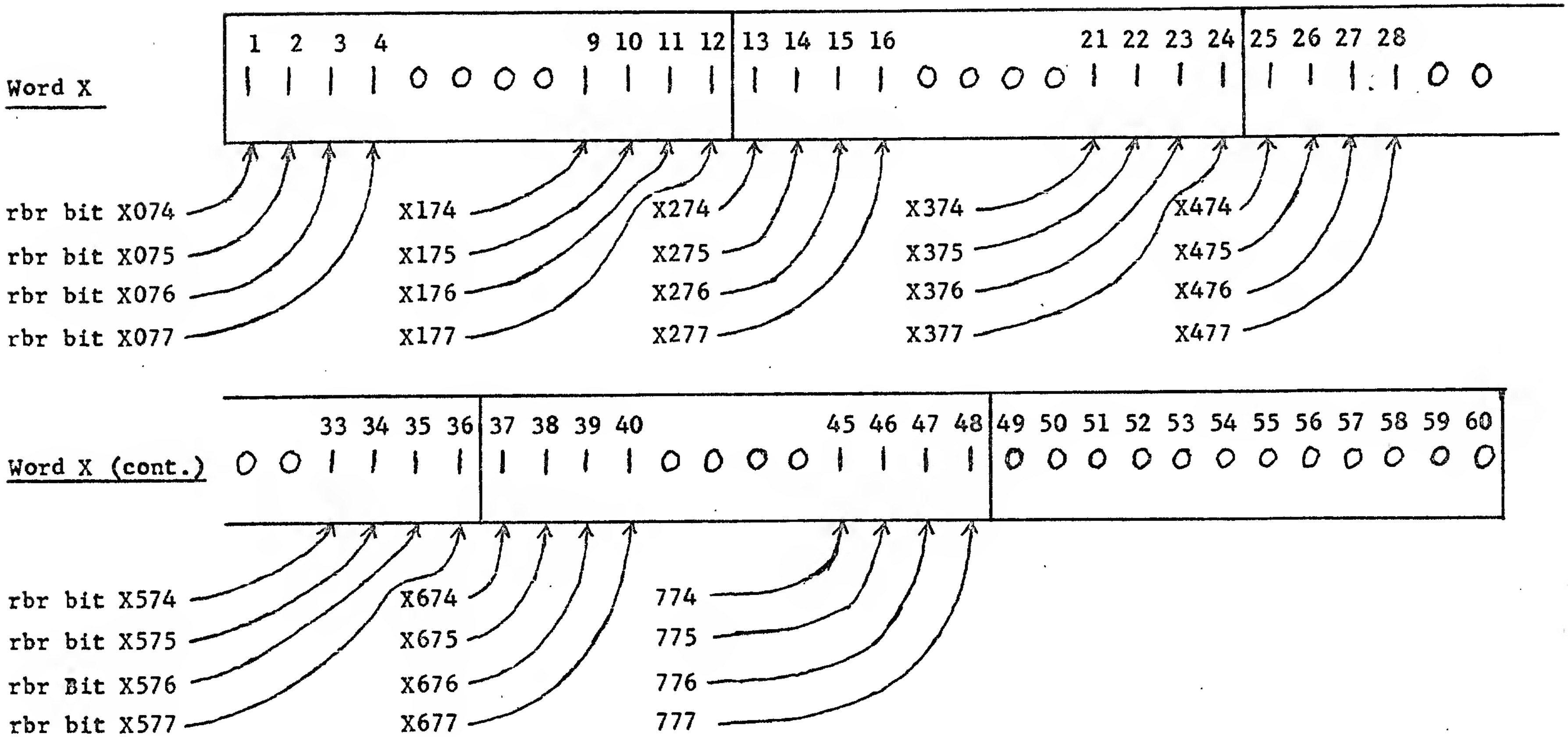
D = Bits 16 - 19 (as usually numbered) or 41 - 44 (as numbered above)

E = Bits 0 - 11 (as usually numbered) or 49 - 60 (as numbered above)

A number corresponding to a bit in the RBR Table may be calculated from and RB number found in one of the 0-7 bytes of an RBT word pair. Suppose the RB number is N (N is always odd). Then the RBR bit number, NRBR, is given by the equation (using truncated integer arithmetic):

$NRBR=N/2=J_u x8^3+J_L x8^2+L_1 x8^1+L_2 x8^0$ (eqn. 1) $J=J_u x8^3+J_L x8^2=$ Main table pointer to words 3-34 (decimal) or 0-37 (octal) of the RBR table. The entire J pointer is used when the last 2 octal digits of NRBR are from 0-73.

DETAIL OF REPRESENTATIVE TAIL WORD



X is 0, 1, 2, or 3, depending on whether we are in tail word 40, 41, 42, or 43.

3-21-6

$L=L_1 \times 8^1+L_2 \times 8^0$=Lower 2 octal digits of RBR bit number. If L is 74-77 (octal) then J_u determines which of the tail words (0-3) the designated RBR bit falls in. Also, J_L determines in which byte in the tail word the bit falls.

For J_L=0 to 1, byte number=1 (counting bytes left to right, 1-5).

For J_L=2 to 3, byte number=2.

For J_L=4 to 5, byte number=3.

For J_L=6 to 7, byte number=4.

A formula for obtaining the exact position of a given bit in one of the tail words (numbering the bits from left to right, 1-60) is given by:

BIT POSITION=LPOSN(J_L+1)+L_2-4 (eqn. 2)

where:

LPOSN(1)=1
LPOSN(2)=9
LPOSN(3)=13
LPOSN(4)=21 These values are decimal.
LPOSN(5)=25
LPOSN(6)=33
LPOSN(7)=37
LPOSN(8)=45

Sample calculation 1:

One of the 0-7 bytes in an RBT word pair contains the RB number 6247. Find the corresponding bit position in the RBR table.

Applying equation 1:

NRBR=6247/2=3123 (dropping the fraction)

L=last 2 digits=23, which is between 0 and 73, and thus the desired bit is in one of the RBR table words from 0 to 37 (octal).

J=31, which points to the 31st (octal) word in the RBR table.

Thus, the desired bit is bit 23 (octal - starting with zero and counting from left to right) of word 31 of the RBR table. The location of this bit is circled on the diagram on Page 12.

Sample calculation 2:

One of the 0-7 bytes in an RBT word pair contains the RB number 3777. Locate the corresponding bit in the RBR table.

Applying equation 1 from page 13:

J_u J_L L_1 L_2

NRBR=3777/2=1777

L=77, which implies that the bit is in one of the tail words.

J_u=1, so the bit is in the second tail word (word 41_8).

J_L=7=second digit from left.

L_2=low order digit=7.

Applying equation 2 from page 14:

BIT POSITION=LPOSN(J_L+1)+L_2-4

BIT POSITION=LPOSN(8)+7-4

BIT POSITION=45+3=48

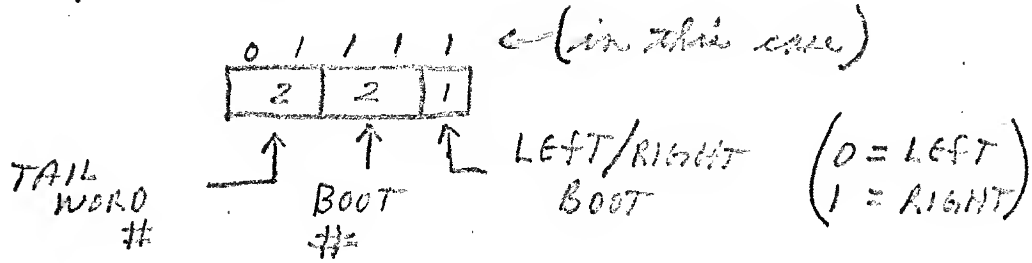
3-21-7

Example # 2.

1. Suppose an RBT byte contained 3777.

2. the actual RB address would be 1777.

3. the lower 6 bits = 77. when they equal 74-77, the next 5 bits are broken down as follows;



4. the lower 2 bits of the lower 6 bits indicate the 2^{xx} position in the boot. In this case 11_2 or the right-most bit of the boot. (Remember they are reserved from left to right).

5. Thus actual RB address # 1777 is tail word # 1, boot 3, right boot, bit 3. or 2^{12} in tail word #1.

Thus, the desired bit is bit 48 (decimal - starting with one and counting from left to right) of the second tail word (word 41). Or, if we count in octal, and start with zero, the bit is number 57 (still counting from left to right). This bit is circled on the diagram on Page 13.

3-21-?

DISCERNING PARITY ERRORS ON DISK:

an indication of a parity Error on Disk is by an entry into the Dayfile and an entry at the associated control point.

Example:

01. 23. 16. ABC0001 . DISK PARITY ERROR

RBR	RB	PRU
01	0741	012

assuming the following configuration —

RBR

00	FILE 0	6638
01	FILE 1	6638

Then —

RBR	RB	PRU
01	0741	012

would be decoded as —

- 6638 file 1
- position 7
- odd half track
- Head group 20_8
- sector 12_8

note — the RB # is actual RB address, not as shown in RBT system.

Physical Position on the Disk

Another question which arises is how one computes the track number, head group, and whether we are writing on odd or even sectors. These quantities may also be derived from the RB numbers found in the 0-7 bytes of the RBT word pairs. The steps are as follow:

1. Assume the RB number is N (12 bits).
2. Divide N by 2 (shift right by 1), leaving us with an eleven bit quantity, X.
3. Divide X up into three groups as indicated below, thus giving us the desired quantities:

X=	7 bits	3 bits	1
	Track	Head Group	Even/Odd Sectors

Thus, the track numbers can go from 0 to 177_8 (0 to 128_{10}) and the head groups can go from 0 to 7. From these numbers, we can calculate the total possible number of half tracks per RBR Table:

Total=(128 tracks/RBR) x (8 head groups) x (2 half-tracks/track)

so:

Total number of half-tracks = 2048

Figures 1-4 on Pages 1-9 of the 6638 manual is somewhat misleading in that it seems to indicate that head groups are specified by two digits. Each RBR Table is for one stack. Thus, if we are going to use only one octal digit to specify a head group, it would seem necessary to supply the File

Unit [illegible] (0 or 1), also. However, this [illegible], [illegible], taken care of by supplying the track number. Thus, there are head groups 0-7 for File Unit 0 and 0-7 for File Unit 1 (for stack 0, as well as stack 1). In order to specify any unique location on one stack, it is sufficient to give:

1. The 3 bit head group number,
2. The 7 bit track number, and
3. The 1 bit even/odd sector indicator (0 for even sectors, 1 for odd sectors).

6638 System for Discerning Parity Errors on Disc File

RBR

00 01	File 1 (Bryant Disk File)
02	File 0 (6638)
03	File 1 (6638)

RB

00 00	Upper 6 (six) bits signify position. Lowest bit signifies odd or even halftrack. The remaining bits will be right shifted one and this will give you the Hd group.

PRU

Signifies the sector #.

Example:	RBR	ACTUAL RB	PRU
	03	0741	012

would be decoded as 6638 File 1
Position 7
odd halftrack
Hd group 20
sector 12

[illegible] 2^{4} [illegible] = 2^{3} [illegible] = 2^{26} [illegible]

DAYFILE BUFFERS

DAYFILE POINTER

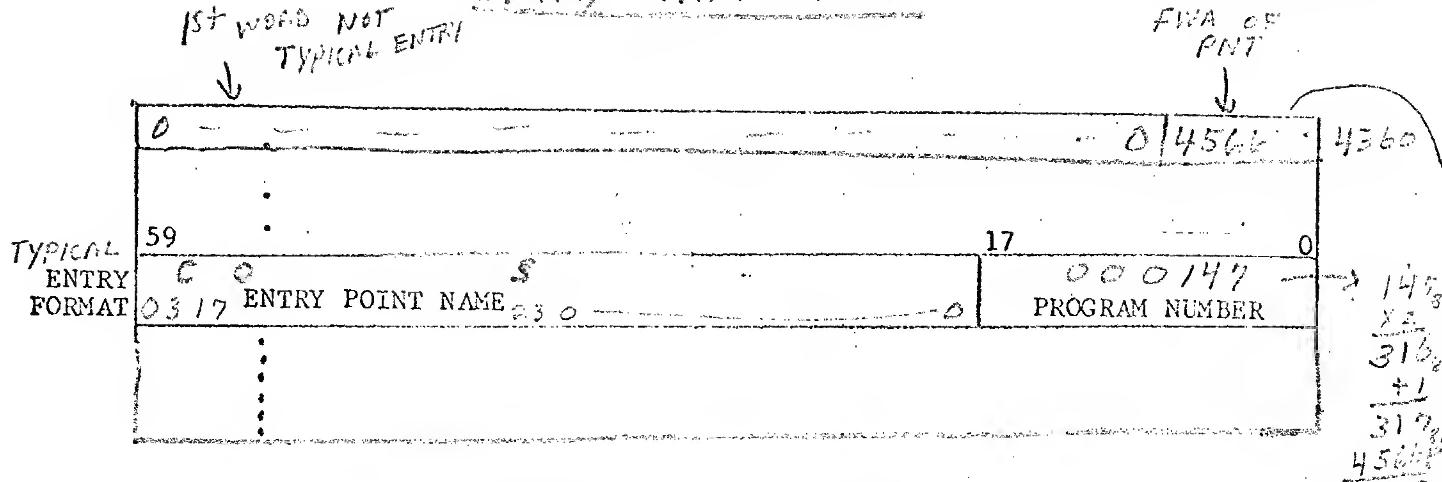
P.DFB	T.DFB/10B	0 ——— 0

DFB

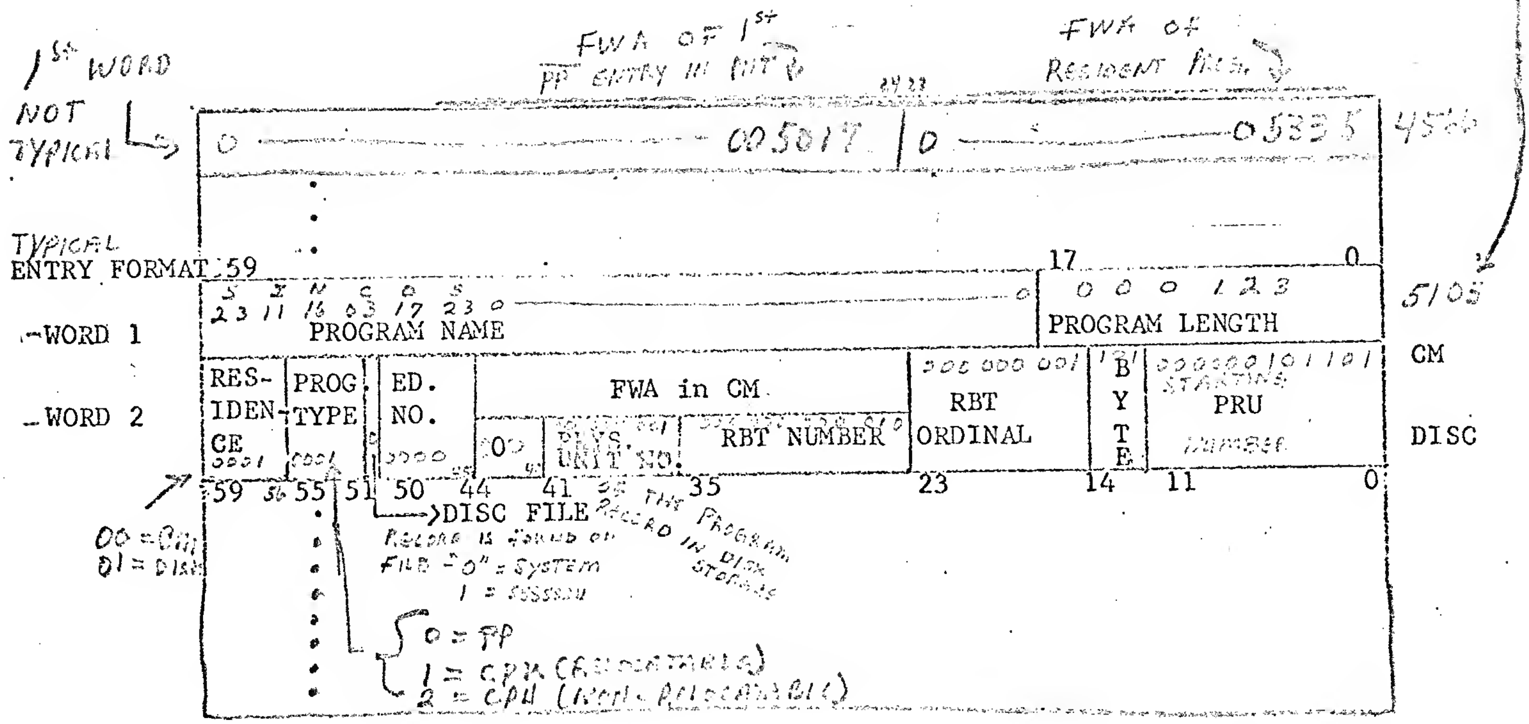
T.DFB	SYSTEM DAYFILE FET (LFN = DAYFILE)	12_8	3140
	CONTROL POINT 1 FET (LFN = DFILE1)	10_8	
	CONTROL POINT 2 FET (LFN = DFILE2)	10_8	
	CONTROL POINT 3 FET (LFN = DFILE3)	10_8	100_8 CM WORDS
	CONTROL POINT 4 FET (LFN = DFILE4)	10_8	
	CONTROL POINT 5 FET (LFN = DFILE5)	10_8	
	CONTROL POINT 6 FET (LFN = DFILE6)	10_8	
	CONTROL POINT 7 FET (LFN = DFILE7)	10_8	3240
	SYSTEM DAYFILE BUFFER	100_8	3340
	⋮		
	CONTROL POINT 7 DAYFILE BUFFER	100_8	

```
THE FORMAT OF THE CMR DIRECTORY IS THIS --
IN CELL P.DIR (I.E. ABSOLUTE 1)
      VFD      24/A,24/B,12/0
WHERE A IS THE ADDRESS OF THE FIRST WORD IN THE DIRECTORY,
AND B IS THE ADDRESS OF THE LAST+1 WORD.
IN CELL A
      VFD      60/C
THEN THE ENTRY POINT TABLE LIES IN CELLS A+1 TO C-1 INCLUSIVE.
EACH ENTRY IN THIS TABLE HAS THE FORM
      VFD      42/D,18/E
WHERE D IS THE NAME OF THE ENTRY POINT, AND E IS THE NUMBER OF
THE PROGRAM, WHICH WILL BE A CP PROGRAM NOT AN OVERLAY, TO WHICH
IT BELONGS. THE PROGRAM NAME TABLE ENTRY FOR THE PROGRAM CAN BE
FOUND IN CELLS C+1+2E AND C+1+2E+1.
IN CELL C
      VFD      36/F,24/G
THEN THE PROGRAM NAME TABLE LIES IN CELLS C+1 TO G-1 INCLUSIVE.
F IS THE ADDRESS OF THE FIRST WORD OF THE FIRST PROGRAM NAME
TABLE ENTRY FOR A NON-PP PROGRAM. THIS HELPS THE PP RESIDENT,
WHICH SEARCHES ONLY FOR PP PROGRAMS IN THE PROGRAM NAME TABLE,
AND NEED SEARCH ONLY FROM C+1 TO F-2
EACH PROGRAM NAME TABLE ENTRY IS TWO WORDS. THE FIRST WORD IS
      VFD      42/H,18/K
WHERE H IS THE NAME OF THE PROGRAM, AND K IS ITS LENGTH --
NOT ITS LENGTH AFTER LOADING, BUT ITS LENGTH IN THE LIBRARY.
K DOES NOT INCLUDE THE PREFIX OF THE PROGRAM RECORD, SO
THIS RECORD ON FILE SYSTEM IS ACTUALLY K+3 WORDS LONG.
BUT IF THE PROGRAM IS CENTRAL MEMORY RESIDENT, ITS BODY IN
THE DIRECTORY WILL BE K WORDS LONG.
NOTE THAT NO TWO NON-STITCH PROGRAMS MAY HAVE THE SAME NAME,
EVEN IF OF DIFFERENT TYPES. BUT STITCH PROGRAMS ARE ABSOLUTELY
IGNORED IN CHECKING FOR DUPLICATION.
THE SECOND WORD OF THE PROGRAM NAME TABLE ENTRY HAS TWO POSSIBLE
FORMATS. IF THE PROGRAM IS CENTRAL MEMORY RESIDENT,
      VFD      4/M,4/N,1/P,6/Q,21/R,9/S,3/W,12/T
WHERE M IS THE RESIDENCE, 0 FOR CENTRAL MEMORY, N IS THE
PROGRAM TYPE, 0 FOR PP, 1 FOR CP, 2 FOR OVERLAY, AND
4 FOR STITCH, P IS THE DISK FILE IN WHICH THE PROGRAM
RECORD IS TO BE FOUND, 0 FOR ((SYSTEM)) AND 1 FOR
((SSSSSSU)), Q IS THE EDITION NUMBER, R IS THE ADDRESS IN
CMR AT WHICH THE BODY OF THE PROGRAM BEGINS
(IT ENDS AT R+K-1), S IS THE RBT ORDINAL OF THE PROGRAM
RECORD IN DISK STORAGE, W IS THE BYTE NUMBER IN THE RBT WORD
PAIR, AND T IS THE PRU NUMBER AT WHICH IT BEGINS IN DISK STORAGE.
IF THE PROGRAM IS DISK RESIDENT,
      VFD      4/M,4/N,1/P,6/Q,3/0,6/U,12/V,9/S,3/W,12/T
WHERE M IS 1 FOR DISK RESIDENCE, N, P, Q, S, T, AND W ARE THE
SAME AS ABOVE, U IS THE PHYSICAL UNIT NUMBER OF THE
PROGRAM RECORD IN DISK STORAGE, AND V IS THE RBT NUMBER,
NOT ORDINAL, OF THE RECORD.
THE PROGRAM NAME TABLE ENDS IWTH THE WORD AT G-1, AND THE BODIES
OF CENTRAL MEMORY RESIDENT PROGRAMS, UNLESS THERE ARE NONE AND
G=B, EXTEND FROM G THROUGH B-1.
```

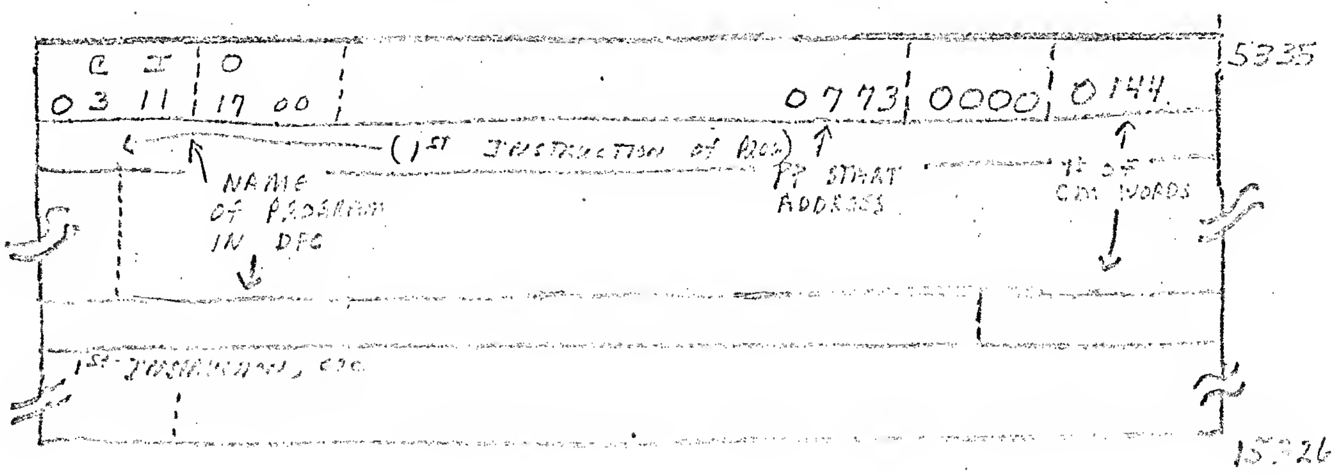
ENTRY POINT TABLE



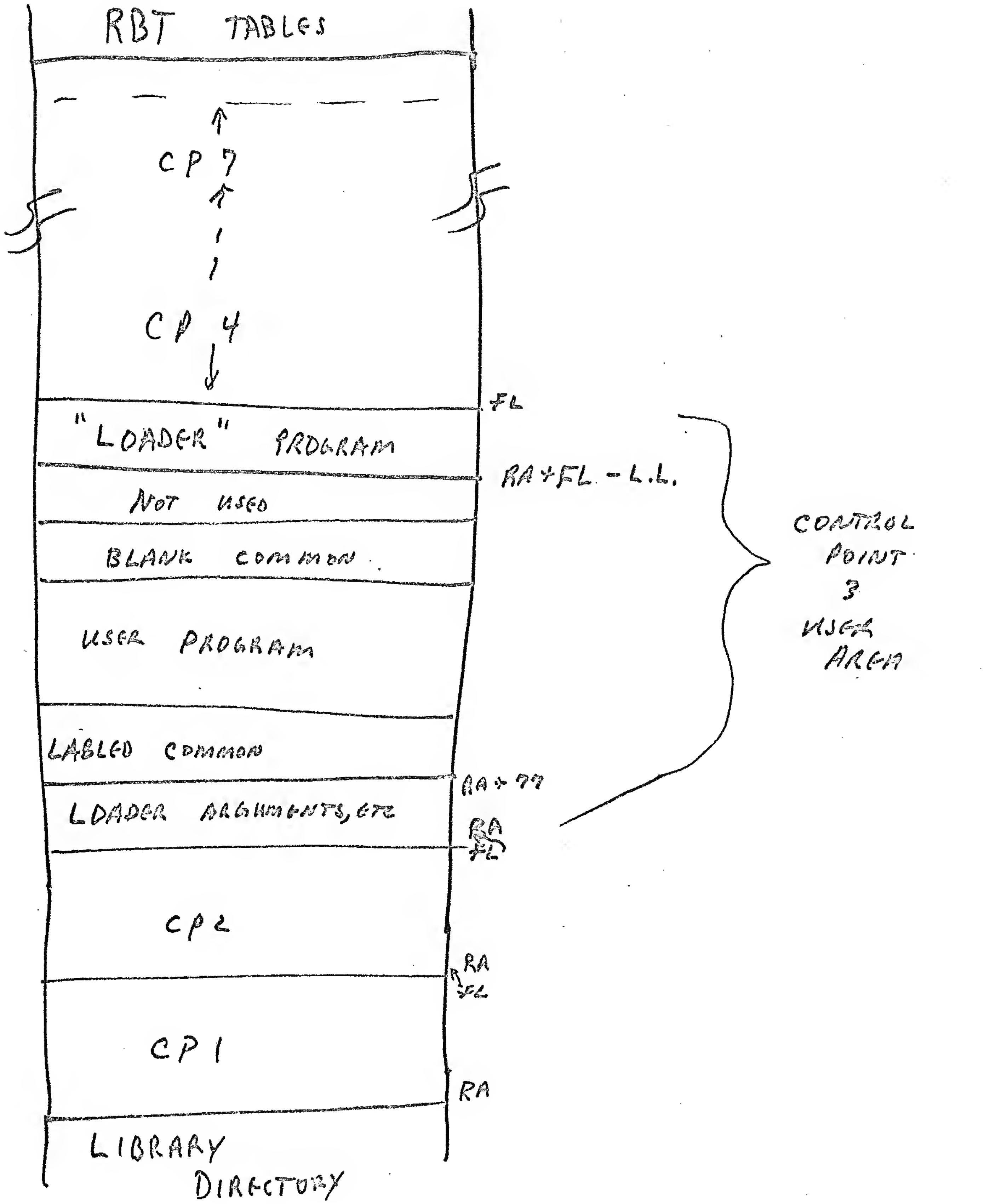
PROGRAM NAME TABLE



CM RESIDENT PROGRAMS



USER SPACE ALLOCATION



Record Block Tables (RBT)

An RBT consists of a series of 12 bit bytes identifying, in logical sequence, the record blocks assigned to a logical file residing on an allocatable device.

A maximum of 8,192 CM words may be occupied by all the RBT's active at any one time. Two CM words at a time are assigned to each RBT as required (Thus, we speak of RBT word pairs); the two words may reference 1 to 8 record blocks (RB's). Two consecutive RBT pairs for a particular file do not have to be contiguous. The first byte of the first word of each RBT pair contains a pointer to the next RBT in the chain. This pointer is the ordinal or relative position of the next RBT. If the pointer is N, the address of the word pair is calculated as:

ADDRESS=(LWA+1 of memory)-2*N=400,000B-2*N.

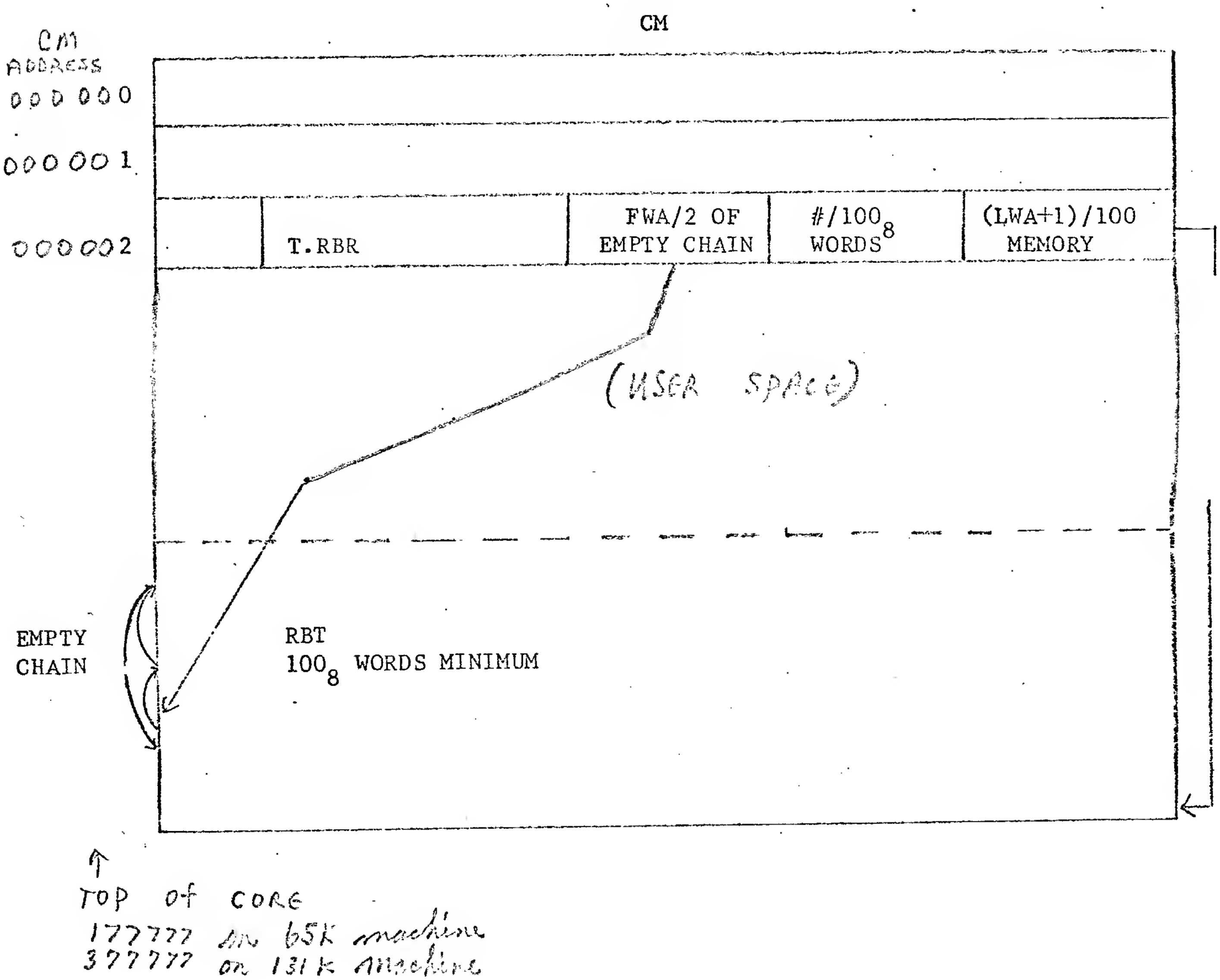
If the pointer is zero, the current RBT ends the chain.

The area reserved for RBT use may be reduced by control point space requirements. The maximum is defined by the number of word pairs referenceable by a twelve bit byte.

The RBT area starts at high core with its first word pair, and works down. The first word of the first RBT is at location 377,776; the first word of the second RBT is at location 377,774; etc. The number of RBT word pairs/100B is found in word 2, byte 3, (starting with 0, left to right) of low core.

The unocccupied RBT pairs are also chained together (blocks of word pairs are added or subtracted in groups of 100B words). The position of the start of the empty chain is found in the middle byte of word 2 of low core.

RBT IN CM



Byte Types That Can Occur in an RBT

RBT Link - (X-byte) The first byte of each RBT CM word pair. If blank, the chain of RBT word pairs assigned to a logical file terminates in the current word pair. If occupied, its contents multiplied by 2 and subtracted from the LWA+1 of central memory gives the address of the next RBT word pair in the chain.

RBR Link - (Y-byte) The high order 9 bits of the 2nd byte of each RBT CM word pair. It identifies the RBR related to the record block group in which the file resides. It may also appear in the third through the tenth byte of each word pair, if the file is contained partially in more than one RB group (We have not found an example of this.).

First RBT Byte - (Y-byte) The first byte used in the RBT word pair is indicated in the low order 3 bits of the 2nd byte of each word pair.

Byte 2 (middle byte) of the first word of a pair is counted as RBT byte 0. Byte 3 is RBT byte 1; byte 4 is RBT byte 2. Byte 0 of word 2 is RBT byte 3, etc. Byte 4 of word 2 is RBT byte 7. See Figure I on Page 25.

If the word pair is the first in an RBT chain the 3-bit "first RBT byte" field (low order 3 bits of the Y-byte) will be set to 2 for sequented files and 4 for random files.

Flag and Allocation Type - (word 1, first pair - "0 byte") Bit 31 contains the release flag; if set, record blocks are to be released after reading.

(RANDOM BIT)
Bit 30 is not clearly defined. (See 2.6.3 (4), Page 26, of SCOPE 3.0 IRS.) Bits 24-29 contain the allocation type as defined in the ERS. 00 and 03 imply no restriction. 01 implies 50 sector record blocks. 02 implies 64 sector RB's.

Next PRU - (bits 12-23, word one - "1 byte") The first word pair in an RBT chain contains the ordinal of the next PRU in which writing is to occur for the referenced logical file. This field may differ from the same field in the FTN, which references the PRU following the last one read or written. This field occurs only in the first word pairs for a logical file.

Last Assigned Record Block - (bits 0-11, word one - "2 byte") - The first word pair for a random file contains the ordinal of the last record block assigned when the file was last written. This field occurs only in the first word pair of a randomly accessed logical file.

Last Assigned PRU - (bits 48-59, word 2 - "3 byte") - The first word pair for a random file contains the ordinal of the last record block assigned when the file was last written. This field occurs only in the first word pair of a randomly accessed logical file.

RB Link - A 12 bit quantity, appearing in one of the 0-7 bytes in an RBT word pair. The upper eleven bits, taken together with the last RBR link, define a unique position on the disk. The lower bit of an RB link is always set to distinguish it from an RBR link (which may also occur in bytes 0-7 when the file is contained partially in more than one RB group). This condition has not been observed yet, however, in our operations at the Los Angeles Data Center. Upon filling one RBR Table, whereupon we expected

the system to continue reserving half-tracks, for all files in the process of being written, in the other RBR Table, the system instead hung up. Dividing the RB link by 2 gives the position in the corresponding RBR Table. (See sample calculations on Page 13.)

See figures on following pages.

RBT WORD PAIR FORMATS

Figure I. SAMPLE WORD PAIR

59	47	35	23	11 0
X-BYTE	Y-BYTE	0-BYTE	1-BYTE	2-BYTE
3-BYTE	4-BYTE	5-BYTE	6-BYTE	7-BYTE

Figure II. FIRST RBT PAIR FOR A SEQUENTIAL FILE

RBT LINK	RBR LINK	BYTE	FLAGS AND ALLOCATION TYPE	NEXT PRU	1ST RB LINK
2ND RB LINK	3RD RB LINK		4TH RB LINK	5TH RB LINK	6TH RB LINK

Figure III. FIRST RBT PAIR FOR A RANDOM FILE

RBT LINK	RBR LINK	BYTE	FLAGS AND ALLOCATION TYPE	NEXT PRU	LAST ASSIGNED RB
LAST ASSIGNED PRU	1ST RB LINK		2ND RB LINK	3RD RB LINK	4TH RB LINK

Figure IV. RBT PAIR INTERMEDIATE IN THE CHAIN (I.E., NOT FIRST OR LAST)

RBT LINK	RBR LINK	0	NTH RB LINK	N+1ST RB LINK	N+2ND RB LINK
N+3RD RB LINK	N+4TH RB LINK		N+5TH RB LINK	N+6TH RB LINK	N+7TH RB LINK

Figure V. LAST RBT PAIR OF A PARTICULAR CHAIN

0 0 0 0	RBR LINK	0	MTH RB LINK	M+1ST RB LINK	M+2ND RB LINK
M+3RD RB LINK	LAST RB LINK		0 0 0 0	0 0 0 0	0 0 0 0

NOTE: In the last example above, the X-BYTE is zero. The LAST RB LINK may fall in any of the bytes 0-7. The bytes after the LAST RB LINK in the last word pair are all zero.

SAMPLE RBT CHAIN

377654	0 0 0 0	0 0 1 0	0 3 4 5	0 3 4 7	0 0 0 0	52 Last Pair of Sample Chain
377655	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	
377704	0 0 5 2	0 0 1 0	0 3 0 1	0 3 0 7	0 3 1 5	36
377705	0 3 1 7	0 3 2 1	0 3 2 3	0 3 3 7	0 3 4 3	
377706	0 0 3 6	0 0 1 0	0 2 0 7	0 2 2 5	0 2 3 1	35
377707	0 2 5 7	0 2 6 1	0 2 6 3	0 2 6 5	0 2 7 7	
377734	0 0 3 5	0 0 1 0	0 1 2 5	0 1 2 7	0 1 3 1	22
377735	0 1 5 5	0 1 5 7	0 2 0 1	0 2 0 3	0 2 0 5	
377736	0 0 2 2	0 0 1 2	0 0 0 3	0 0 0 4	0 0 3 7	21 First Pair of Sample Chain
377737	0 0 4 1	0 0 5 7	0 0 6 3	0 1 0 1	0 1 2 3	

Designates RBR Table 01

Means file is sequential because first RB LINK of first word pair starts in the 2-byte.

The RBR Table for "System" is 00.

"System" is a random file.

377776	0 0 0 2	0 0 0 4				1 First Pair of "System" Chain
377777						

HYPOTHETICAL FNT/FST ENTRY
ASSOCIATED WITH RBT CHAIN ON
PREVIOUS PAGE

59		47	35	23		11	0
FILE NAME				TYPE	C.P.	PRIORITY	
EQP. CODE	CK	FIRST RBT PAIR	CURRENT RBT PAIR	CURRENT RBT ENTRY	BYTE	SECTOR	
	FET ADDRESS		DISPOSITION CODE	SEC. CODE	LAST CODE AND STATUS		

The designations of the FNT/FST field contents are repeated here for convenience. See pages ff. for details.

Assume that we have rewound the file after writing it and then have read up to RB LINK 0337, SECTOR 52. (This is in word pair 36, byte 6.) The FNT/FST entry might look like that below:

T	A	P	E	1				
24	01	20	05	34	0000	3	2	0000

0 2 0 0	0 0 2 1	0 0 3 6	0 0 3	6	0 0 5 2	
0 0	0 0 2 2 7 6	0 0 0 0	0 2	0 0 0 0 1 1		

TAPE 1

From the FNT/FST entry, we can get the following information:

1. The file name is TAPE 1.
2. The lock bit is not set.
3. The type is local (03).
4. The control point is 2.
5. There is no priority associated with the file at this time.
6. The equipment code specifies 6638 Disk, alternate sector half-tracks (02).
7. The first RBT pair is the 21st in the table.

8. The current RBT pair (where file is currently positioned) is 36B.
9. The current RBT entry is 3, i.e., there are 3 completely full RBT word pairs up to (preceding) the current pair in which the file is positioned (the current position is in word pair array 36B).
10. The current byte position in RBT pair 36 is 6.
11. The sector number is 52.
12. The address of the start of the File Environment Table (FET) associated with this file is 2276.
13. There is no disposition code associated with this file.
14. The security code is 02.
15. The last code and status is 11, which says the last operation completed was a read.

Pertinent Disk Tables

The FNT/FST area, RBT (Record Block Table) word pair area, and RBR (Record Block Reservation) tables play important roles in the disk reservation scheme. Low core contains pointers to and information related to these tables.

On the diagrams on the following pages:

A = Pointer to the start of the FNT/FST area.

B = Pointer to the LWA+1 of the FNT/FST area.

C = Length of RBT area/100B. The address of the first word of the RBT area = 400 000B-C*100B.

D = Start of RBR tables area (=B in the current LADC system).

E = Start of the empty chain in the RBT word pairs area (ie., the ordinal of the first word pair of the empty chain). The address associated with the first word of this word pair is 400 000B-2*E.

F = The contents of the second byte of the second word of an FNT/FST entry. F is the ordinal of the first RBT word pair for that file.

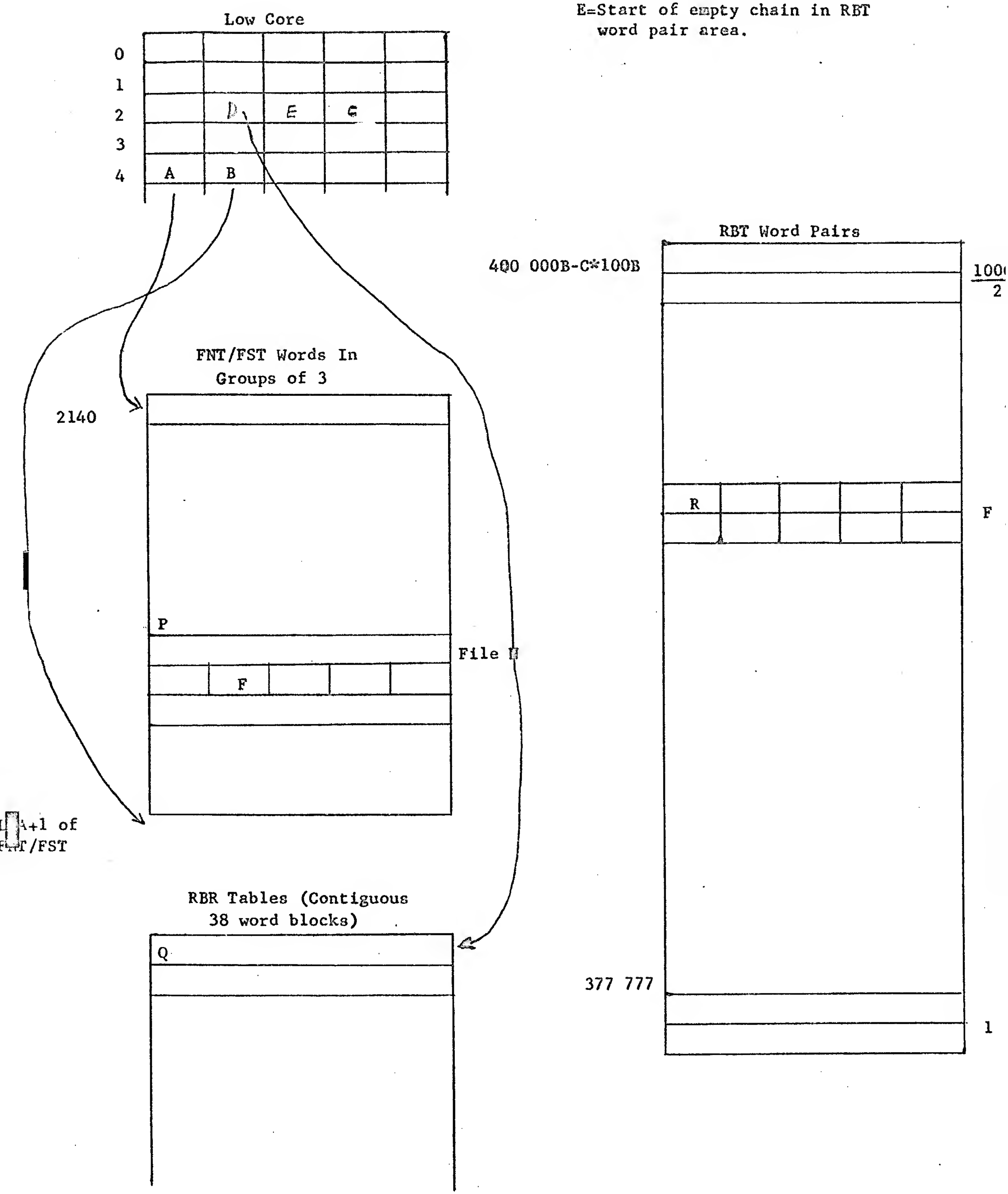
P - Refers to a 3 word FNT/FST entry.

Q - Refers to a 38 word RBR table.

R - Refers to an RBT word pair.

POINTERS AND TABLES

C=Length of RBT area/100B.
E=Start of empty chain in RBT
word pair area.



Program Disk

Purpose

To perform on-line dynamic verification of the SCOPE 3.1 Disk Linkage. Reservation tables or the corresponding tables that are placed in an image area after each dead start by the bootstrap programs L99 and BOO.

Method

This program prints out 20 words of low core, the FNT/FST entries, the RBT area, RBR tables, and the RBT empty chain. It then checks to be sure that there is a bit set in the appropriate RBR Table corresponding to each RBR link in the RBT area. An error message is printed out for each discrepancy found. Each FNT/FST entry is printed out followed by the proper disk number and all of the RB links associated with that file. The final result of the test is displayed on the scope and also goes to the dayfile.

Usage

1. To verify the current DISK Tables - DISK.
2. To verify the IMAGE DISK Tables - DISK (IMAGE)

Restrictions

The CM routines MIORC, MIORF, MBYTE, LBYT, MEMGET, and MEMPUT, and the PP routine MIO are used.

The program currently assumes two disks (2 RBR tables).

Sample Output:

LOW CORE AREA

000000	0000	0000	0000	0000	0000
000001	0000	6700	0003	1036	0000
000002	0000	4420	0027	0003	4000
000003	0560	0000	0000	0000	1206
000004	2140	4420	0000	0000	0000
000005	2040	2140	0000	0000	0000
000006	0000	0000	0000	0000	0003
000007	5340	0000	0000	0000	0000
000010	0000	0000	0000	0000	0000
000011	0000	0000	0000	0000	0000
000012	0000	0000	0000	0000	0000
000013	0276	0000	2262	0002	0454
000014	0000	0000	0011	0000	0000
000015	0000	0000	0000	0001	0000
000016	0000	0000	0000	0000	0000
000017	0000	0000	0000	0000	0000

FNT/FST AREA

```
002140     0401 3106 1114 0530 0000     DAYFILEX
002141     0000 0041 0041 0002 0007        6 6 B G
002142     0000 0000 0000 0000 0016              N

002143     2331 2324 0515 0020 0000     SYSTEM P
002144     0200 0001 0001 0004 0000     B  A A D
002145     0000 0626 0000 0000 0053       FV     S

002146     0406 1114 0534 0030 0000     DFILE1 X
002147     0000 0114 0114 0002 0005       ALAL B E
002150     0000 0000 0000 0000 0017              O

002151     0406 1114 0535 0030 0000     DFILE2 X
002152     0000 0000 0000 0002 0000            B
002153     0000 0010 0000 0074 1073        H   ≤H>

002154     0406 1114 0536 0030 0000     DFILE3 X
002155     0000 0000 0000 0002 0000            B
002156     0000 0010 0000 0074 1073        H   ≤H>

002157     0406 1114 0537 0030 0000     DFILE4 X
002160     0000 0026 0026 0002 0001        V V B A
002161     0000 0010 0000 0000 0017        H     O

002162     0406 1114 0540 0030 0000     DFILE5 X
002163     0000 0000 0000 0002 0000            B
002164     0000 0010 0000 0074 1073        H   ≤H>

002165     0406 1114 0541 0030 0000     DFILE6 X
002166     0000 0000 0000 0002 0000            B
002167     0000 0010 0000 0074 1073        H   ≤H>

002170     0406 1114 0542 0030 0000     DFILE7 X
002171     0000 0000 0000 0000 0000
002172     0000 0000 0000 0000 0001              A

002173     2205 0104 0100 0031 0000     READA  Y
002174     6000 0000 0000 0000 0000     ≡
002175     0000 0010 0000 0000 0031        H     Y

002176     2205 0104 0300 0031 0000     READC  Y
002177     6000 0000 0000 0000 0000     ≡
002200     0000 0030 0000 0000 0031        X     Y

002201     1116 2025 2400 0034 0000     INPUT  1
002202     0200 0021 0021 0002 0053     B  Q Q B S
002203     0000 0100 0000 0000 0021        A     Q

002204     0130 3132 0000 0020 0000     AXYZ   P
002205     0200 0020 0020 0003 0060     B  P P C ≡
002206     0000 2112 0000 0200 0025       QJ   B U

002207     1407 1700 0000 0034 0000     LGO    1
002210     0200 0022 0022 0002 0003     B  R R B C
002211     0000 0626 0000 0200 0023       FV   B S

002212     2022 1116 2403 0011 0003     PRINTC I C
002213     0200 0025 0030 0011 0026     B  U X I V
```

22

3-39

```
002215    1725 2420 2524 0034 0000    OUTPUT 1
002216    0200 0023 0023 0004 0025    B  S S D U
002217    0000 0114 0000 0200 0025      AL  B  U

002220    0315 2023 0322 0034 0000    CMPSCR 1
002221    0000 0000 0000 0000 0000
002222    0000 3022 0000 0000 0051      XR       (

002223    2323 2323 2323 2634 0000    SSSSSSV1
002224    0000 0024 0024 0002 0001        T T B A
002225    0000 0100 0000 0000 0021      A        Q

002226    2323 2323 2323 3034 0000    SSSSSSX1
002227    0000 0000 0000 0000 0000
002230    0000 0635 0000 0000 0053      F2       $

002234    2323 2323 2323 2420 0000    SSSSSSTP
002235    0000 0034 0034 0004 0056        1 1 D .
002236    0000 0626 0000 0000 0027      FV       W

002237    2323 2323 2323 2520 0000    SSSSSSUP
002240    0000 0040 0040 0006 0057        5 5 F .
002241    0000 0626 0000 0000 0027      FV       W
```

CURRENT LENGTH OF RRT AREA = N = 0300B

377500	0000	0000	0022	0023	0000	0140
377501	0000	0000	0000	0000	0000	
377502	0140	0000	0022	0023	0000	0137
377503	0000	0000	0000	0000	0000	
377504	0137	0000	0022	0023	0000	0136
377505	0000	0000	0000	0000	0000	
377506	0136	0000	0022	0023	0000	0135
377507	0000	0000	0000	0000	0000	
377510	0135	0000	0000	0000	0000	0134
377511	0000	0000	0000	0000	0000	
377512	0134	0000	0000	0000	0000	0133
377513	0000	0000	0000	0000	0000	
377514	0133	0000	0000	0000	0000	0132
377515	0012	0012	0012	0012	0012	
377516	0132	0000	0000	0000	0000	0131
377517	0002	0002	0002	0002	0000	
377520	0131	0000	0000	0000	0000	0130
377521	0002	0002	0002	0002	0002	
377522	0130	0000	0000	0000	0000	0127
377523	0002	0002	0002	0002	0002	
377524	0127	0000	0000	0000	0000	0126
377525	0000	0000	0000	0000	0000	
377526	0126	0000	0000	0000	0000	0125
377527	0002	0002	0002	0002	0002	
377530	0125	0000	0000	0000	0000	0124
377531	0002	0002	0002	0002	0002	
377532	0124	0000	0000	0000	0000	0123
377533	0002	0002	0000	0000	0000	
377534	0123	0000	0000	0000	0000	0122
377535	0002	0002	0002	0002	0002	
377536	0122	0000	0000	0000	0000	0121
377537	0002	0002	0002	0002	0002	
377540	0121	0000	0000	0000	0000	0120
377541	0002	0002	0002	0002	0002	
377542	0120	0000	0000	0000	0000	0117
377543	0002	0002	0002	0002	0002	
377544	0117	0000	0000	0000	0000	0116
377545	0012	0000	0000	0000	0000	

3-41

377546	0116	0000	0000	0000	0000	0115
377547	0002	0002	0002	0002	0002	
377550	0000	0002	0000	0005	0667	0114
377551	0000	0000	0000	0000	0000	
377552	0115	0000	0000	0000	0000	0113
377553	0000	0000	0000	0000	0000	
377554	0113	0000	0000	0000	0000	0112
377555	0012	0012	0012	0012	0012	
377556	0112	0000	0000	0000	0000	0111
377557	0012	0012	0012	0012	0012	
377560	0111	0000	0000	0000	0000	0110
377561	0012	0012	0012	0012	0012	
377562	0110	0000	0000	0000	0000	0107
377563	0012	0012	0012	0012	0012	
377564	0107	0000	0000	0000	0000	0106
377565	0012	0012	0012	0012	0012	
377566	0106	0000	0000	0000	0000	0105
377567	0012	0012	0012	0012	0012	
377570	0105	0000	0000	0000	0000	0104
377571	0002	0002	0002	0002	0002	
377572	0104	0000	0000	0000	0000	0103
377573	0000	0000	0000	0000	0000	
377574	0103	0000	0000	0000	0000	0102
377575	0012	0012	0012	0012	0012	
377576	0102	0000	0000	0000	0000	0101
377577	0012	0012	0012	0012	0012	
377600	0101	0000	0000	0000	0000	0100
377601	0012	0012	0012	0012	0012	
377602	0100	0000	0000	0000	0000	0077
377603	0012	0012	0012	0012	0012	
377604	0077	0000	0000	0000	0000	0076
377605	0012	0012	0012	0012	0012	
377606	0076	0000	0000	0000	0000	0075
377607	0000	0000	0000	0000	0000	
377610	0075	0000	0000	0000	0000	0074
377611	0002	0002	0000	0000	0000	
377612	0074	0000	0000	0000	0000	0073
377613	0002	0002	0002	0002	0002	
377614	0073	0000	0000	0000	0000	0072
377615	0002	0002	0002	0002	0002	
377616	0072	0000	0000	0000	0000	0071
377617	0002	0002	0002	0002	0002	
377620	0071	0000	0000	0000	0000	0070
377621	0002	0002	0002	0002	0002	

35

3-42

377622	0070	0000	0000	0000	0000	0067
377623	0012	0000	0000	0000	0000	
377624	0067	0000	0000	0000	0000	0066
377625	0002	0002	0002	0002	0002	
377626	0066	0000	0000	0000	0000	0065
377627	0002	0002	0002	0002	0002	
377630	0065	0000	0000	0000	0000	0064
377631	0002	0002	0002	0002	0002	
377632	0064	0000	0000	0000	0000	0063
377633	0002	0002	0002	0002	0002	
377634	0063	0000	0000	0000	0000	0062
377635	0002	0002	0002	0002	0002	
377636	0062	0000	0000	0000	0000	0061
377637	0002	0002	0002	0002	0002	
377640	0061	0000	0000	0000	0000	0060
377641	0002	0002	0002	0002	0002	
377642	0060	0000	0000	0000	0000	0057
377643	0002	0002	0002	0000	0000	
377644	0057	0000	0000	0000	0000	0056
377645	0002	0002	0002	0002	0002	
377646	0056	0000	0000	0000	0000	0055
377647	0000	0000	0000	0000	0000	
377650	0055	0000	0000	0000	0000	0054
377651	0000	0000	0000	0000	0000	
377652	0054	0000	0000	0000	0000	0053
377653	0002	0002	0002	0002	0002	
377654	0053	0000	0000	0000	0000	0052
377655	0000	0000	0000	0000	0000	
377656	0052	0000	0000	0000	0000	0051
377657	0000	0000	0000	0000	0000	
377660	0051	0000	0000	0000	0000	0050
377661	0012	0012	0012	0012	0012	
377662	0050	0000	0000	0000	0000	0047
377663	0012	0012	0012	0012	0012	
377664	0047	0000	0000	0000	0000	0046
377665	0012	0012	0000	0000	0000	
377666	0046	0000	0000	0000	0000	0045
377667	0012	0012	0012	0012	0012	
377670	0045	0000	0000	0000	0000	0044
377671	0012	0012	0012	0012	0012	
377672	0044	0000	0000	0000	0000	0043
377673	0012	0012	0012	0012	0012	
377674	0043	0000	0000	0000	0000	0042
377675	0000	0000	0000	0000	0000	

3-43

377676	0000	0002	0000	0007	0365	0041
377677	0000	0000	0000	0000	0000	
377700	0000	0014	0101	0057	0004	0040
377701	0000	0055	0071	0133	0000	
377702	0042	0000	0000	0000	0000	0037
377703	0012	0012	0012	0012	0012	
377704	0037	0000	0000	0000	0000	0036
377705	0000	0000	0000	0000	0000	
377706	0036	0000	0000	0000	0000	0035
377707	0012	0012	0012	0012	0012	
377710	0000	0012	0003	0056	0073	0034
377711	0045	0047	0000	0000	0000	
377712	0035	0000	0000	0000	0000	0033
377713	0012	0012	0012	0012	0012	
377714	0000	0010	0065	0063	0067	0032
377715	0000	0000	0000	0000	0000	
377716	0033	0000	0000	0000	0000	0031
377717	0000	0000	0000	0000	0000	
377720	0032	0010	0027	0033	0035	0030
377721	0041	0043	0051	0053	0057	
377722	0031	0000	0000	0000	0000	0027
377723	0000	0000	0000	0000	0000	
377724	0000	0002	0000	0001	0003	0026
377725	0000	0000	0000	0000	0000	
377726	0030	0012	0003	0051	0011	0025
377727	0015	0017	0021	0023	0025	
377730	0000	0012	0003	0001	0037	0024
377731	0000	0000	0000	0000	0000	
377732	0000	0012	0003	0025	0005	0023
377733	0007	0013	0000	0000	0000	
377734	0000	0012	0003	0004	0061	0022
377735	0000	0000	0000	0000	0000	
377736	0000	0012	0003	0053	0001	0021
377737	0000	0000	0000	0000	0000	
377740	0000	0012	0003	0060	0003	0020
377741	0031	0000	0000	0000	0000	
377742	0000	0000	0335	0337	0341	0017
377743	0343	0000	0000	0000	0000	
377744	0017	0000	0315	0317	0321	0016
377745	0323	0325	0327	0331	0333	
377746	0016	0000	0275	0277	0301	0015
377747	0303	0305	0307	0311	0313	
377750	0015	0000	0255	0257	0261	0014
377751	0263	0265	0267	0271	0273	

377752	0014	0000	0235	0237	0241	0013
377753	0243	0245	0247	0251	0253	
377754	0013	0000	0215	0217	0221	0012
377755	0223	0225	0227	0231	0233	
377756	0012	0000	0175	0177	0201	0011
377757	0203	0205	0207	0211	0213	
377760	0011	0000	0155	0157	0161	0010
377761	0163	0165	0167	0171	0173	
377762	0010	0000	0135	0137	0141	0007
377763	0143	0145	0147	0151	0153	
377764	0007	0000	0115	0117	0121	0006
377765	0123	0125	0127	0131	0133	
377766	0006	0000	0075	0077	0101	0005
377767	0103	0105	0107	0111	0113	
377770	0005	0000	0055	0057	0061	0004
377771	0063	0065	0067	0071	0073	
377772	0004	0000	0035	0037	0041	0003
377773	0043	0045	0047	0051	0053	
377774	0003	0000	0015	0017	0021	0002
377775	0023	0025	0027	0031	0033	
377776	0002	0004	0100	0024	0163	0001
377777	0024	0005	0007	0011	0013	

RBR TABLE 00

004420	0002	0100	0000	0301	0100
004421	4000	0001	4000	3614	0000

004422	7777	7777	7777	7777	7777
004423	7777	7777	7777	7777	6002
004424	0000	0000	0000	0000	0000
004425	0000	0000	0400	0000	0000
004426	0000	0000	0000	0000	0000
004427	0000	0000	0000	0000	0000
004430	0000	0000	0000	0000	0000
004431	0000	0000	0000	0000	0000
004432	0000	0000	0000	0000	0000
004433	0000	0000	0000	0000	0000
004434	0000	0000	0000	0000	0000
004435	0000	0000	0000	0000	0000
004436	0000	0000	0000	0000	0000
004437	0000	0000	0000	0000	0000
004440	0000	0000	0000	0000	0000
004441	0000	0000	0000	0000	0000
004442	0000	0000	0000	0000	0000
004443	0000	0000	0000	0000	0000
004444	0000	0000	0000	0000	0000
004445	0000	0000	0000	0000	0000
004446	0000	0000	0000	0000	0000
004447	0000	0000	0000	0000	0000
004450	0000	0000	0000	0000	0000
004451	0000	0000	0000	0000	0000
004452	0000	0000	0000	0000	0000
004453	0000	0000	0000	0000	0000
004454	0000	0000	0000	0000	0000
004455	0000	0000	0000	0000	0000
004456	0000	0000	0000	0000	0000
004457	0000	0000	0000	0000	0000
004460	0000	0000	0000	0000	0000
004461	0000	0000	0000	0000	0000

004462	7400	0000	0000	0000	0000
004463	0000	0000	0000	0000	0000
004464	0000	0000	0000	0000	0000
004465	0000	0000	0000	0000	0000

RBR TABLE 01

004466	0002	0200	0000	0301	0100
004467	4000	0002	4000	3741	0002

004470	7777	7777	7700	0004	0000
004471	0000	0000	0000	0000	0000
004472	0000	0000	0000	0000	0000
004473	0000	0000	0000	0000	0000
004474	0000	0000	0000	0000	0000
004475	0000	0000	0000	0000	0000
004476	0000	0000	0000	0000	0000
004477	0000	0000	0000	0000	0000
004500	0000	0000	0000	0000	0000
004501	0000	0000	0000	0000	0000
004502	0000	0000	0000	0000	0000
004503	0000	0000	0000	0000	0000
004504	0000	0000	0000	0000	0000
004505	0000	0000	0000	0000	0000
004506	0000	0000	0000	0000	0000
004507	0000	0000	0000	0000	0000
004510	0000	0000	0000	0000	0000
004511	0000	0000	0000	0000	0000
004512	0000	0000	0000	0000	0000
004513	0000	0000	0000	0000	0000
004514	0000	0000	0000	0000	0000
004515	0000	0000	0000	0000	0000
004516	0000	0000	0000	0000	0000
004517	0000	0000	0000	0000	0000
004520	0000	0000	0000	0000	0000
004521	0000	0000	0000	0000	0000
004522	0000	0000	0000	0000	0000
004523	0000	0000	0000	0000	0000
004524	0000	0000	0000	0000	0000
004525	0000	0000	0000	0000	0000
004526	0000	0000	0000	0000	0000
004527	0000	0000	0000	0000	0000

004530	0000	0000	0000	0000	0000
004531	0000	0000	0000	0000	0000
004532	0000	0000	0000	0000	0000
004533	0000	0000	0000	0000	0000

```
FN   T CONTENTS WITH ASSOCIATED RECORD BLOCKS FROM RBT   LE


002140      0401 3106 1114 0530 0000      DAYFILEX
002141      0000 0041 0041 0002 0007         6 6 B G
002142      0000 0000 0000 0000 0016               N

            THE FOLLOWING RBS ARE FOR DISK 00

            0365

002143      2331 2324 0515 0020 0000      SYSTEM P
002144      0200 0001 0001 0004 0000      B  A A D
002145      0000 0626 0000 0000 0053        FV     $

            THE FOLLOWING RBS ARE FOR DISK 00

            0005 0007 0011 0013 0015 0017 0021 0023 0025 0027 0031 0033 0035 0037 0041 0043 0045 0047 0051 0053

            0055 0057 0061 0063 0065 0067 0071 0073 0075 0077 0101 0103 0105 0107 0111 0113 0115 0117 0121 0123

            0125 0127 0131 0133 0135 0137 0141 0143 0145 0147 0151 0153 0155 0157 0161 0163 0165 0167 0171 0173

            0175 0177 0201 0203 0205 0207 0211 0213 0215 0217 0221 0223 0225 0227 0231 0233 0235 0237 0241 0243

            0245 0247 0251 0253 0255 0257 0261 0263 0265 0267 0271 0273 0275 0277 0301 0303 0305 0307 0311 0313

            0315 0317 0321 0323 0325 0327 0331 0333 0335 0337 0341 0343

002146      0406 1114 0534 0030 0000      DFILE1 X
002147      0000 0114 0114 0002 0005        ALAL B E
002150      0000 0000 0000 0000 0017               O

            THE FOLLOWING RBS ARE FOR DISK 00

            0667

002151      0406 1114 0535 0030 0000      DFILE2 X
002152      0000 0000 0000 0002 0000             B
002153      0000 0010 0000 0074 1073          H   <H>

      THERE ARE NO RBT WORD PAIRS FOR THE ABOVE FNT/FST ENTRY

002154      0406 1114 0536 0030 0000      DFILE3 X
002155      0000 0000 0000 0002 0000             B
002156      0000 0010 0000 0074 1073          H   <H>

      THERE ARE NO RBT WORD PAIRS FOR THE ABOVE FNT/FST ENTRY

002157      0406 1114 0537 0030 0000      DFILE4 X
002160      0000 0026 0026 0002 0001          V V B A
002161      0000 0010 0000 0000 0017          H     O

            THE FOLLOWING RBS ARE FOR DISK 00
```

```
            0003                                                    41
                                                                    3-48

002162      0406 1114 0540 0030 0000      DFILE5 X
002163      0000 0000 0000 0002 0000             B
002164      0000 0010 0000 0074 1073          H  <H>

    THERE ARE NO RBT WORD PAIRS FOR THE ABOVE FNT/FST ENTRY


002165      0406 1114 0541 0030 0000      DFILE6 X
002166      0000 0000 0000 0002 0000             B
002167      0000 0010 0000 0074 1073          H  <H>

    THERE ARE NO RBT WORD PAIRS FOR THE ABOVE FNT/FST ENTRY


002170      0406 1114 0562 0030 0000      DFILE7 X
002171      0000 0000 0000 0000 0000
002172      0000 0000 0000 0000 0001                A

    THERE ARE NO RBT WORD PAIRS FOR THE ABOVE FNT/FST ENTRY


002173      2205 0104 0100 0031 0000      READA  Y
002174      6000 0000 0000 0000 0000      ≡
002175      0000 0010 0000 0000 0031         H     Y

    THIS FNT/FST ENTRY IS FOR A NON-ALLOCATABLE DEVICE

    THERE ARE NO RBT WORD PAIRS FOR THE ABOVE FNT/FST ENTRY


002176      2205 0104 0300 0031 0000      READC  Y
002177      6000 0000 0000 0000 0000      ≡
002200      0000 0030 0000 0000 0031         X     Y

    THIS FNT/FST ENTRY IS FOR A NON-ALLOCATABLE DEVICE

    THERE ARE NO RBT WORD PAIRS FOR THE ABOVE FNT/FST ENTRY


002201      1116 2025 2400 0034 0000      INPUT  1
002202      0200 0021 0021 0002 0053      B  Q Q B S
002203      0000 0100 0000 0000 0021        A      Q

                THE FOLLOWING RBS ARE FOR DISK 01

                0001


002204      0130 3132 0000 0020 0000      AXYZ   P
002205      0200 0020 0020 0003 0060      B  P P C ≡
002206      0000 2112 0000 0200 0025        QJ  B  U

                THE FOLLOWING RBS ARE FOR DISK 01

                0003 0031


002207      1407 1700 0000 0034 0000      LGO    1
002210      0200 0022 0022 0002 0003      B  R R B C
002211      0000 0626 0000 0200 0023        FV  B  S
```

3-49

```
002212      2022 1116 2403 0011 0003      PRINTC I C
002213      0200 0025 0030 0011 0026      B  U X I V
002214      0000 0030 0040 0200 0011        X 5B  I

            THE FOLLOWING RBS ARE FOR DISK 01

            0011 0015 0017 0021 0023 0025 0027 0033 0035 0041 0043 0051 00

002215      1725 2420 2524 0034 0000      OUTPUT 1
002216      0200 0023 0023 0004 0025      B  S S D U
002217      0000 0114 0000 0200 0025        AL  B  U

            THE FOLLOWING RBS ARE FOR DISK 01

            0005 0007 0013

002220      0315 2023 0322 0034 0000      CMPSCR 1
002221      0000 0000 0000 0000 0000
002222      0000 3022 0000 0000 0051        XR     (

     THERE ARE NO RBT WORD PAIRS FOR THE ABOVE FNT/FST ENTRY

002223      2323 2323 2323 2634 0000      SSSSSSV1
002224      0000 0024 0024 0002 0001        T T B A
002225      0000 0100 0000 0000 0021        A      Q

            THE FOLLOWING RBS ARE FOR DISK 01

            0037

002226      2323 2323 2323 3034 0000      SSSSSSX1 .
002227      0000 0000 0000 0000 0000
002230      0000 0635 0000 0000 0053        F2     $

     THERE ARE NO RBT WORD PAIRS FOR THE ABOVE FNT/FST ENTRY

002234      2323 2323 2323 2420 0000      SSSSSSTP
002235      0000 0034 0034 0004 0056        1 1 D .
002236      0000 0626 0000 0000 0027        FV     W

            THE FOLLOWING RBS ARE FOR DISK 01

            0073 0045 0047

002237      2323 2323 2323 2520 0000      SSSSSSUP
002240      0000 0040 0040 0006 0057        5 5 F .
002241      0000 0626 0000 0000 0027        FV     W

            THE FOLLOWING RBS ARE FOR DISK 01

            0055 0071 0133
```

THE RBT PAIRS IN THE EMPTY CHAIN FOLLOW

```
377722     0031 0000 0000 0000 0000     0027
377723     0000 0000 0000 0000 0000

377716     0033 0000 0000 0000 0000     0031
377717     0000 0000 0000 0000 0000

377712     0035 0000 0000 0000 0000     0033
377713     0012 0012 0012 0012 0012

377706     0036 0000 0000 0000 0000     0035
377707     0012 0012 0012 0012 0012

377704     0037 0000 0000 0000 0000     0036
377705     0000 0000 0000 0000 0000

377702     0042 0000 0000 0000 0000     0037
377703     0012 0012 0012 0012 0012

377674     0043 0000 0000 0000 0000     0042
377675     0000 0000 0000 0000 0000

377672     0044 0000 0000 0000 0000     0043
377673     0012 0012 0012 0012 0012

377670     0045 0000 0000 0000 0000     0044
377671     0012 0012 0012 0012 0012

377666     0046 0000 0000 0000 0000     0045
377667     0012 0012 0012 0012 0012

377664     0047 0000 0000 0000 0000     0046
377665     0012 0012 0000 0000 0000

377662     0050 0000 0000 0000 0000     0047
377663     0012 0012 0012 0012 0012

377660     0051 0000 0000 0000 0000     0050
377661     0012 0012 0012 0012 0012

377656     0052 0000 0000 0000 0000     0051
377657     0000 0000 0000 0000 0000

377654     0053 0000 0000 0000 0000     0052
377655     0000 0000 0000 0000 0000

377652     0054 0000 0000 0000 0000     0053
377653     0002 0002 0002 0002 0002

377650     0055 0000 0000 0000 0000     0054
377651     0000 0000 0000 0000 0000

377646     0056 0000 0000 0000 0000     0055
377647     0000 0000 0000 0000 0000

377644     0057 0000 0000 0000 0000     0056
```

44

3-51

377642	0060	0000	0000	0000	0000	0057
377643	0002	0002	0002	0000	0000	
377640	0061	0000	0000	0000	0000	0060
377641	0002	0002	0002	0002	0002	
377636	0062	0000	0000	0000	0000	0061
377637	0002	0002	0002	0002	0002	
377634	0063	0000	0000	0000	0000	0062
377635	0002	0002	0002	0002	0002	
377632	0064	0000	0000	0000	0000	0063
377633	0002	0002	0002	0002	0002	
377630	0065	0000	0000	0000	0000	0064
377631	0002	0002	0002	0002	0002	
377626	0066	0000	0000	0000	0000	0065
377627	0002	0002	0002	0002	0002	
377624	0067	0000	0000	0000	0000	0066
377625	0002	0002	0002	0002	0002	
377622	0070	0000	0000	0000	0000	0067
377623	0012	0000	0000	0000	0000	
377620	0071	0000	0000	0000	0000	0070
377621	0002	0002	0002	0002	0002	
377616	0072	0000	0000	0000	0000	0071
377617	0002	0002	0002	0002	0002	
377614	0073	0000	0000	0000	0000	0072
377615	0002	0002	0002	0002	0002	
377612	0074	0000	0000	0000	0000	0073
377613	0002	0002	0002	0002	0002	
377610	0075	0000	0000	0000	0000	0074
377611	0002	0002	0000	0000	0000	
377606	0076	0000	0000	0000	0000	0075
377607	0000	0000	0000	0000	0000	
377604	0077	0000	0000	0000	0000	0076
377605	0012	0012	0012	0012	0012	
377602	0100	0000	0000	0000	0000	0077
377603	0012	0012	0012	0012	0012	
377600	0101	0000	0000	0000	0000	0100
377601	0012	0012	0012	0012	0012	
377576	0102	0000	0000	0000	0000	0101
377577	0012	0012	0012	0012	0012	
377574	0103	0000	0000	0000	0000	0102
377575	0012	0012	0012	0012	0012	
377572	0104	0000	0000	0000	0000	0103
377573	0000	0000	0000	0000	0000	
377570	0105	0000	0000	0000	0000	0104
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	

65

2-52

377566	0106	0000	0000	0000	0000	0105
377567	0012	0012	0012	0012	0012	
377564	0107	0000	0000	0000	0000	0106
377565	0012	0012	0012	0012	0012	
377562	0110	0000	0000	0000	0000	0107
377563	0012	0012	0012	0012	0012	
377560	0111	0000	0000	0000	0000	0110
377561	0012	0012	0012	0012	0012	
377556	0112	0000	0000	0000	0000	0111
377557	0012	0012	0012	0012	0012	
377554	0113	0000	0000	0000	0000	0112
377555	0012	0012	0012	0012	0012	
377552	0115	0000	0000	0000	0000	0113
377553	0000	0000	0000	0000	0000	
377546	0116	0000	0000	0000	0000	0115
377547	0002	0002	0002	0002	0002	
377544	0117	0000	0000	0000	0000	0116
377545	0012	0000	0000	0000	0000	
377542	0120	0000	0000	0000	0000	0117
377543	0002	0002	0002	0002	0002	
377540	0121	0000	0000	0000	0000	0120
377541	0002	0002	0002	0002	0002	
377536	0122	0000	0000	0000	0000	0121
377537	0002	0002	0002	0002	0002	
377534	0123	0000	0000	0000	0000	0122
377535	0002	0002	0002	0002	0002	
377532	0124	0000	0000	0000	0000	0123
377533	0002	0002	0000	0000	0000	
377530	0125	0000	0000	0000	0000	0124
377531	0002	0002	0002	0002	0002	
377526	0126	0000	0000	0000	0000	0125
377527	0002	0002	0002	0002	0002	
377524	0127	0000	0000	0000	0000	0126
377525	0000	0000	0000	0000	0000	
377522	0130	0000	0000	0000	0000	0127
377523	0002	0002	0002	0002	0002	
377520	0131	0000	0000	0000	0000	0130
377521	0002	0002	0002	0002	0002	
377516	0132	0000	0000	0000	0000	0131
377517	0002	0002	0002	0002	0000	
377514	0133	0000	0000	0000	0000	0132
377515	0012	0012	0012	0012	0012	
377512	0134	0000	0000	0000	0000	0133

3-53

377510	0135	0000	0000	0000	0000	0134
377511	0000	0000	0000	0000	0000	
377506	0136	0000	0022	0023	0000	0135
377507	0000	0000	0000	0000	0000	
377504	0137	0000	0022	0023	0000	0136
377505	0000	0000	0000	0000	0000	
377502	0140	0000	0022	0023	0000	0137
377503	0000	0000	0000	0000	0000	
377500	0000	0000	0022	0023	0000	0140
377501	0000	0000	0000	0000	0000	

LLOWING ARE THE RESULTS OF A TEST TO CHECK IF THERE IS AN RBR BIT SET FOR EACH RB INDICATED IN THE RBT WORD PAIRS TABLE.

```
R TEST INITIATED FOR FILE DAYFILEX
R TEST INITIATED FOR FILE SYSTEM P
R TEST INITIATED FOR FILE DFILE1 X
R TEST INITIATED FOR FILE DFILE4 X
R TEST INITIATED FOR FILE INPUT  1
R TEST INITIATED FOR FILE AXYZ   P
R TEST INITIATED FOR FILE LGO    1
R TEST INITIATED FOR FILE PRINTC I C
R TEST INITIATED FOR FILE OUTPUT 1
R TEST INITIATED FOR FILE SSSSSSV1
R TEST INITIATED FOR FILE SSSSSSTP
R TEST INITIATED FOR FILE SSSSSSUP
```

9 FILES HAD NO RBT WORD PAIRS ASSOCIATED WITH THEM

12 FILES TESTED FOR RB - RBR CORRESPONDENCE

HE NUMBER OF FILES IN THE FNT/FST AT THE TIME OF TESTING WAS 21

) ERRORS IN RB - RBR CORRESPONDENCE TEST.

PP RESIDENT

D.Z0	0000	
D.Z1		
D.Z2		
D.Z3		
D.Z4		
D.Z5		
D.Z6		
D.Z7		TEMPORARY STORAGE
D.T0	0010	
D.T1		
D.T2		OUTPUT BUFFER
D.T3		
D.T4		
D.T5		
D.T6		
D.T7		
D.TW0/D.FNT	0020	
D.TW1		
D.TW2		
D.TW3		FNT (2) AND TNT (3)
D.TW4		
D.TW5		
D.TW6		
D.TW7		

D.TH0		
D.TH1		
D.TH2/D.EST	0032	EST
D.TH3		
D.TH4		
D.TH5		
D.TH6		
D.TH7/D.DTS/D.JFL		0037
D.FR0/D.BA	0040	FET ADDRESS RELATIVE TO LOWER 18 BITS OF I.R.
D.FR1		
D.FR2		
D.FR3		
D.FR4		
D.FR5/D.JECS	0045	ECS FIELD LENGTH REQUIREMENT
D.FR6/D.JPR		COMPUTING PRIORITY
D.FR7/D.JTL/	10_8	JOB TIME LIMIT
D.FF0/D.PPIRB	0050	INPUT REGISTER BUFFER
D.FF1		
D.FF2		
D.FF3		
D.FF4		
D.FF5/D.RA	0055	CENTRAL MEMORY RA/100B
D.FF6/D.FL		CENTRAL MEMORY FL/100B
D.FF7/D.FA		FILE FST ADDRESS (FNT(2) ADDRESS)

D.SX0/D.FIRST	0060	
D.SX1		
D.SX2/D.IN	0062	
D.SX3		
D.SX4/D.OUT	0064	
D.SX5		
D.SX6/D.LIMIT	0066	
D.SX7		
D.SV0/D.PPONE	0070	1
D.SV1/D.HN	0071	100B
D.SV2/D.TH	0072	1000B
D.SV3/D.TR	0073	3
D.SV4/D.CPAD	0074	CONTROL POINT ADDRESS
D.SV5/D.PPIR	0075	INPUT REGISTER ADDRESS
D.SV6/D.PPOR	0076	OUTPUT REGISTER ADDRESS
D.SV7/D.PPMES1	0077	MESSAGE BUFFER ADDRESS

R.IDLE 0100	IDLE LOOP
R.OVLJ 0111	LOAD A PRIMARY OVERLAY & JUMP TO IT
R.OVL 124	LOAD OVERLAYS
R.EREQS 0300	ENTER A REQUEST IN STACK
R.WAIT 0410	WAIT FOR OUTPUT REG. CLEAR
R.PAUSE 0430	PAUSE FOR CM MOVE
R.MTR 0450 (R.PROCESS)	ISSUE A MONITOR REQUEST
R.READP 0460	DISK I/O ROUTINES
R.WRITEP 0470	
R.STBMSK 0611	MASK FOR R.STB = 7700B
R.STB 0620	MASK BITS IN A LIST OF WORDS

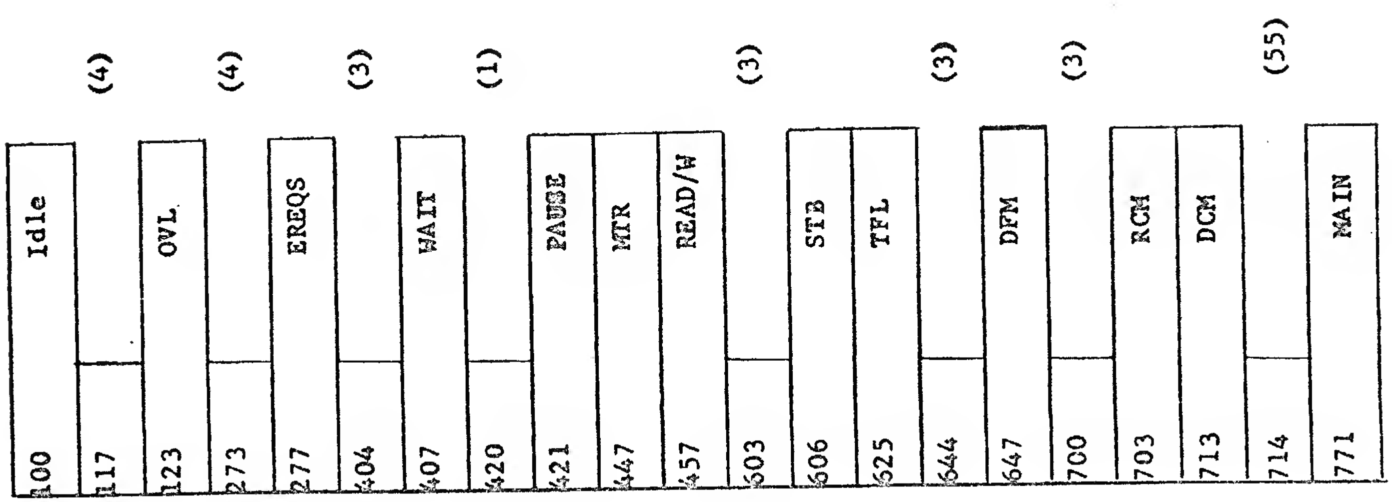
4-1-5

R.CPFL	0627	CONTROL POINT FIELD LENGTH/100_8
R.CPRA	0631	CONTROL POINT RELATIVE ADDRESS/100_8
R.TFL	0634	CHECK ADDRESS FOR IN RANGE
R.DFM	0650	ISSUE DAYFILE MESSAGE
R.RCH	0704	REQUEST CHANNEL
R.DCH	0714	DROP CHANNEL
	0772	
	1772	

Reset by B. PAUSE

PP RESIDENT ENTRY POINTS

Comfile Tag	3.0	3.1	Function
R. IDLE	100	100	Idle loop
R.OVLJ	115	111	Enter to load overlay at 1000
R.OVL	124	124	Load PP overlay
R.EREQS	300	300	Enter request stack
R.WAIT	410	410	Wait output register clear
R.PAUSE	430	430	Pause for relocation
R.PROCESS(R.MTR)	450	450	Issue MTR function
R.READP	460	460	Xmit data to stack processor
R.WRITEP	470	470	Get data from stack processor
R.STBMSK	611	611	Channel table mask
R.STB	620	620	Process channel table mods
R.CPFL	627	627	FL
R.CPRA	631	631	RA
R.TFL	634	634	Add RA to ACC, test vs FL
R.DFM	650	650	Process dayfile message
R.RCH	704	704	Reserve channel
R.DCH	714	714	Drop channel
R.STEP	734	-	



PP RESIDENT

Resident Idle Loop — R.IDLE

Calling Sequence

LJM R.IDLE

R.IDLE is the idle loop; PP resident continually scans its input register for something to do.

Overlay Load — R.OVL

Calling Sequence

Load A register	Load Address
RJM	R.OVL

R.OVL causes an overlay whose name appears in D.T6 & D.T7 (right justified) to be loaded into the PP beginning at the address specified in the A register. R.OVL is used both by PP overlays to load higher level overlays & by PP resident to load the overlay named in the input register. PP resident does not reference the disk directly to load disk resident overlays but makes a call to the stack processor.

Enter Request Stack — R.EREQS

Calling Sequence

Store	L (request) in D.TO
RJM	R.EREQS

R.EREQS adds the control point number to the already formatted

4-2

request & searches the central memory request stack for an empty entry. The monitor function, M.EREQS, is called & PP resident iterates until the monitor accepts the request.

Wait

R.WAIT

Calling Sequence

```
RJM     R.WAIT
```

R.WAIT will cause the PP to idle until the output register is clear.

Pause

R.PAUSE

Calling Sequence

```
RJM     R.PAUSE
STD     D.RA
```

R.PAUSE will exit if the PP is attached to control point zero or if the storage move flag is not set. Otherwise, the monitor function, M.PAUSE, will be issued & the PP will pause until monitor has completed the storage move for that control point. In any event, before an exit is made from R.PAUSE, the following information will be set:

(D.TO + C.CPST)	=	control point status
(D.TO + C.CPEF)	=	control point error flag
(D.TO + C.CPRA)	=	control point RA (hundreds)
(D.TO + C.CPFL)	=	control point FL (hundreds)
A register	=	control point RA

D.RA should always be reset after a jump to R.PAUSE.

<u>Process Monitor Function</u> — <u>R.MTR</u>

Calling Sequence

Store function parameters in D.T1 to D.T4

Load function code

```
RJM     R.MTR
```

R.MTR places the function code in D.T0, writes D.T0 through D.T4 to the output register, & waits for the output register to clear.

<u>Transmit Data Via Channel From/To Stack Processor</u> — <u>R.READP/R.WRITEP</u>

Calling Sequence

```
Load     L (request)
RJM      R.READP/R.WRITEP
```

R.READP/R.WRITEP computes the PP word count from the first & last word addresses given in the already formatted request & adds the computed word count, the address of the PP message buffer, & the control point number to the request. The request is entered in the stack & data is transmitted via channel directly to/from PP memory. Upon exit from R.READP/R.WRITEP, the following information will be set:

(D.T3 + C.RWPPLW)	=	LWA + 1 of data transmitted
(D.T3 + C.RWPPST)	=	status
(D.T3 + C.RWPPWT)	=	number of PP words transmitted

Calling Sequence

Load L (list)

RJM R.STB

where "list" has the form

L (byte)
L (word 1)
L (word 2)
:
L (word n)
zero

An entry point to R.STB called R.STBMSK is the address of the mask "ended" with each word in the list before the word is "exclusive ored" with the byte. This mask is initially 7700B and this value should be restored by any routine which substitutes an alternate mask. R.STB is used primarily to substitute channel numbers in driver overlays.

Test Field Length R.TFL

Calling Sequence

Load relative address

RJM R.TFL

R.TFL is used to insure that a relative address is within the field length. The 18-bit address is added to the control point reference address (RA) & compared with the field length. If the address is out of range, R.TFL will exit with a negative A register; if the

address is legal, the A register will contain the absolute CM address (RA + relative address) upon exit. The control point RA & FL are kept locally within PP resident at R.CPRA & R.CPFL, respectively; these locations are initialized when an entry to R.PAUSE is made.

Enter Dayfile Message — R.DFM

Calling Sequence

Load	L (message) + flag bits
RJM	R.DFM

R.DFM will cause a message to be written from PP memory to the dayfile and/or the console. The flag bits are contained in the high order 6-bits of the A register upon entry to R.DFM & are used to determine the destination(s) of the message. Possible values of the flag bits are described below; one or more bits may be on; all are optional.

1	=	Dayfile only ("A" display)
2	=	control point 0 (system) message
4	=	no "A" display

Drop Channel — R.DCH

Calling Sequence

Load	channel number
RJM	R.DCH

R.DCH will cause the specified channel to be dropped.

Request Channel

R.RCH

Calling Sequence

Load	channel number
RJM	R.RCH

The channel number(s) contained in the "A" register will be stored in byte D.T1, monitor function M.RCH inserted in D.T0, and D.T0 - D.T4 written to the output register for that PP. Channels will be assigned by MTR on the following priority basis:

D.T0	D.T1	D.T2	D.T3	D.T4
	2 , 1	4 , 3		

If alternate channels are specified MTR will stop looking for alternate channels upon sensing 6 bits of zero. Thus if desiring one alternate channel the programmer must clear D.T2 before entering R.RCH so the search will be terminated at that point. The procedure for requesting channel 12 with alternate channel 13 would be:

```
LDN     0
STD     D.T2
LDC     1312B
RJM     R.RCH
```

Monitor will stop looking for alternate channels after 4 channels have been investigated.

When R.RCH is used, D.T4 is automatically set nonzero; in this case, the function is not considered complete (i.e., output register is

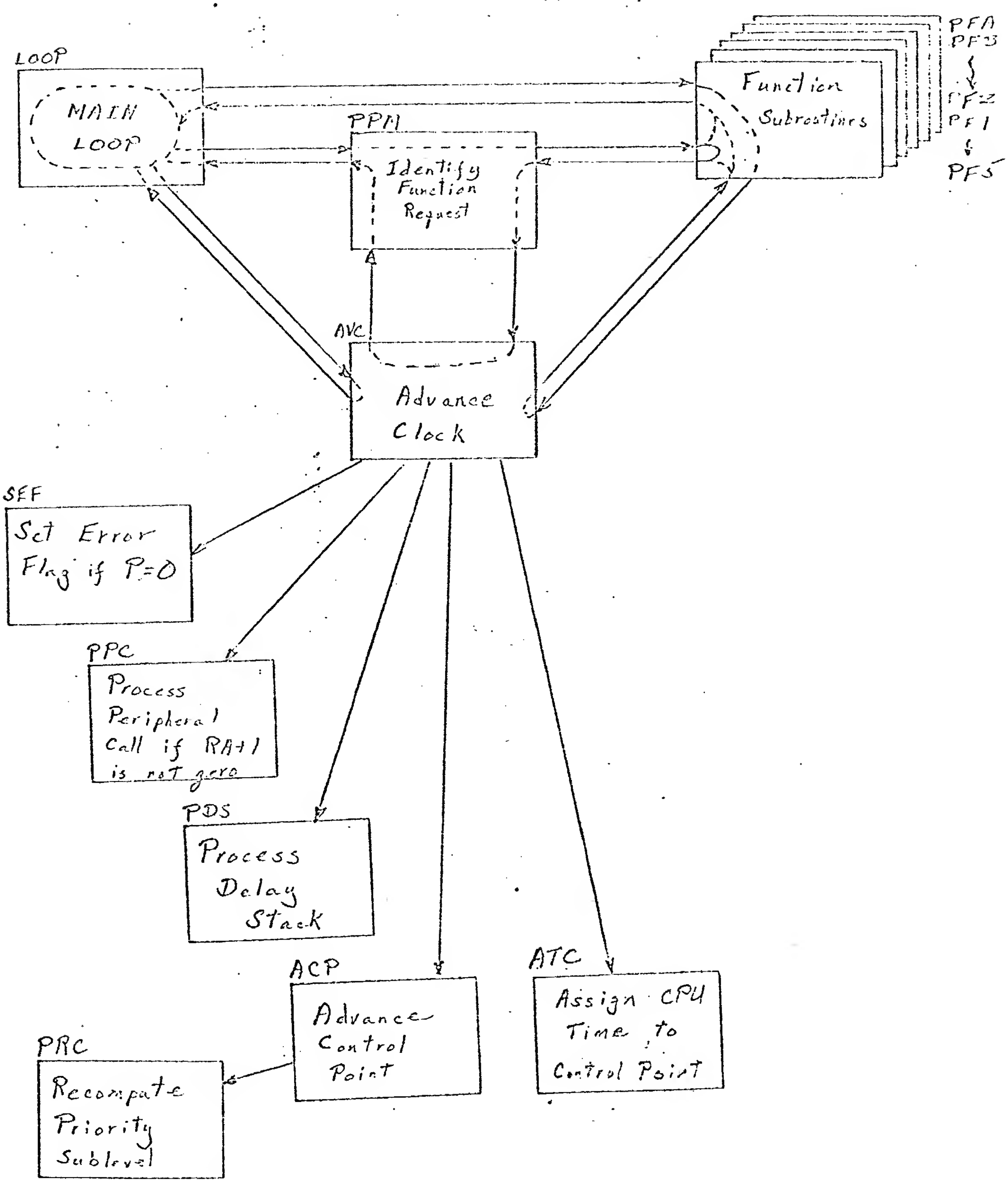
4-7

not cleared) until a channel can be assigned. When complete, byte 0 of the output register is cleared & byte 4 is set to 7777B. A channel request may be made directly to the monitor (M.RCH); one other option is allowed in this case. If byte 4 of the output contains zero, the monitor will notify the requesting PP whether or not the channel could be assigned.

Peripheral Services Request Codes

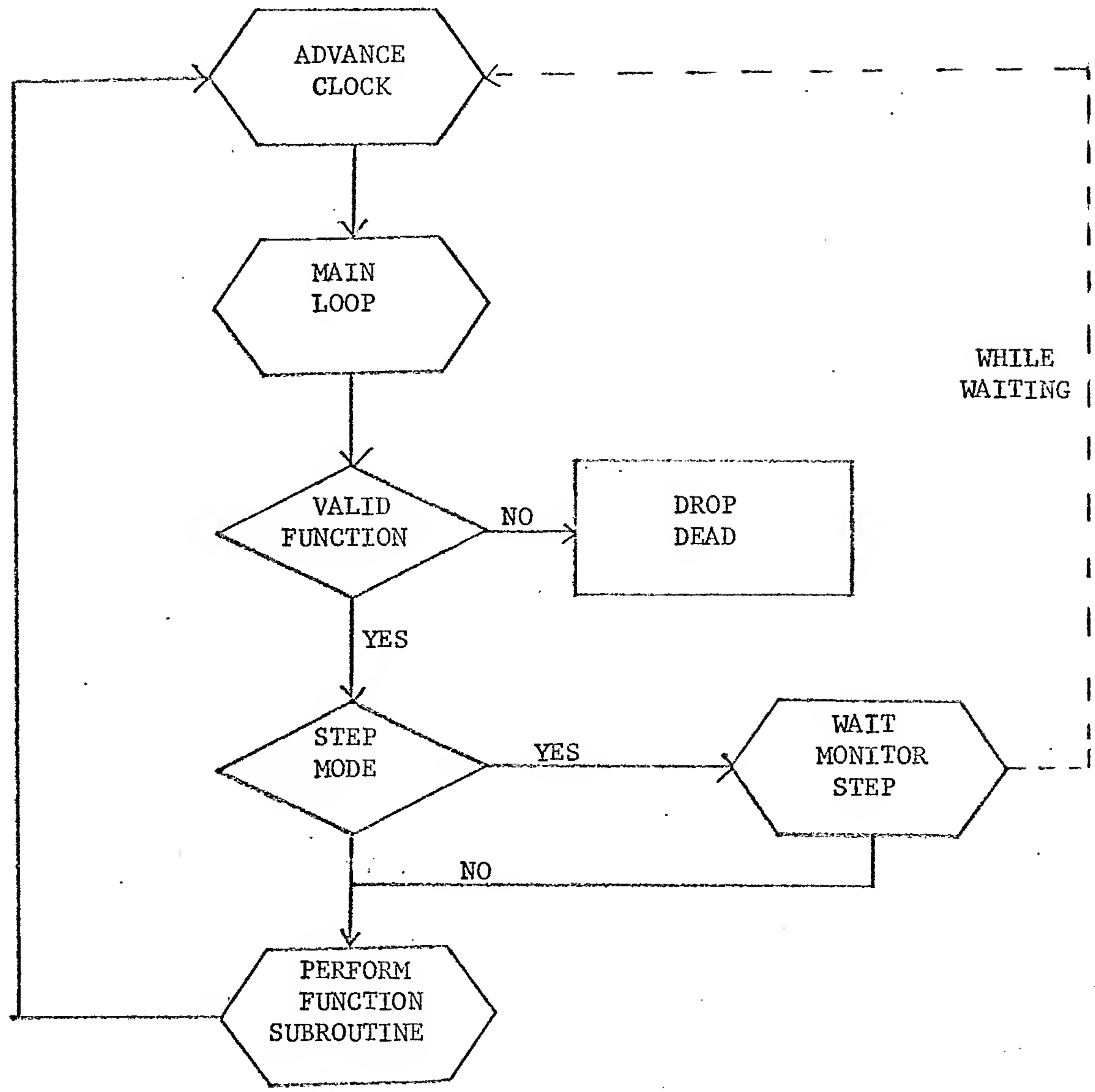
OPCODE	MNEMONIC	DESCRIPTION	OUTPUT REGISTER
01	M.DFM	Dayfile Message	0001 000X 0000 0000 0000 X=0 Normal X=1 Dayfile Only X=2 CPO Message X=4 No A Display
02	M.RCH	Request Chanel	0002 BBAA CCDD 0000 ZZZZ ZZZZ = 2 Hang Till Free AA = Primary Choice BB = Second Choice DD = Third Choice CC = Fourth Choice
03	M.DCH	Drop Chanel	0003 00XX 0000 0000 0000
04	M.PPTIME	Assign Time	0004 0000 0000 0000 0000
05	M.STEP	STEP	0005 0000 0000 0000 000
06			
07			
10	M.RSTOR	Request Storage	0010 FFFF 0000 00XX 0000 XX=10 request RBT Storage(CP8) (BYTE 2 = Low Unit)
11	M.CDF	Complete Dayfile	0011 0000 0000 0000 0000
12	M.DPP	Drop PP	0012 0000 0000 0000 0000
13	M.ABORT	Abort	0013 0000 0000 0000 0000
14	M.NTIME	Time Limit	0014 TTTT 0000 0000 0000
15	M.RCP	Request CP	0015 0000 0000 0000 0000
16	M.DCP	Drop CP	0016 0000 0000 0000 0000
17	M.PAUSE	Pause	0017 0000 0000 0000 0000
20	M.RPP	Request PP	0020 0000 0000 0000 0000
21	M.RCLCP	Recall CP	0021 0000 0000 0000 0000
22	M.REQP	Request EQP	0022 EEEE 0000 0000 0000
23	M.DEQP	Drop EQP	0023 00EE 0000 0000 0000
24	M.RPRI	Priority	0024 00PP 0000 0000 0000
25	M.REM	Exit Mode	0025 000M 0000 0000 0000
26			
27			
30	M.OPDROP	Operator Drop	0030 00CP 00EE 0000 0000
31	M.RTAPE	ON	0031 00EE 0000 0000 0000
32	M.DTAPE	OFF	0032 00EE 0000 0000 0000
33	M.AEQP	Assign EQPT	0033 00EE 000P 0000 0000
34	M.EREQS	Enter Request	0034 000R 00AA SSSS 0000
35	M.CCPA	Change CP	0035 0000 0000 0000 00CP
36	M.CPUST	Change CPY	0036 0000 0000 0000 0000
37	M.RPJ	Delay PP Call	0037 MMMM MMMM bbbb 0000

— MONITOR —

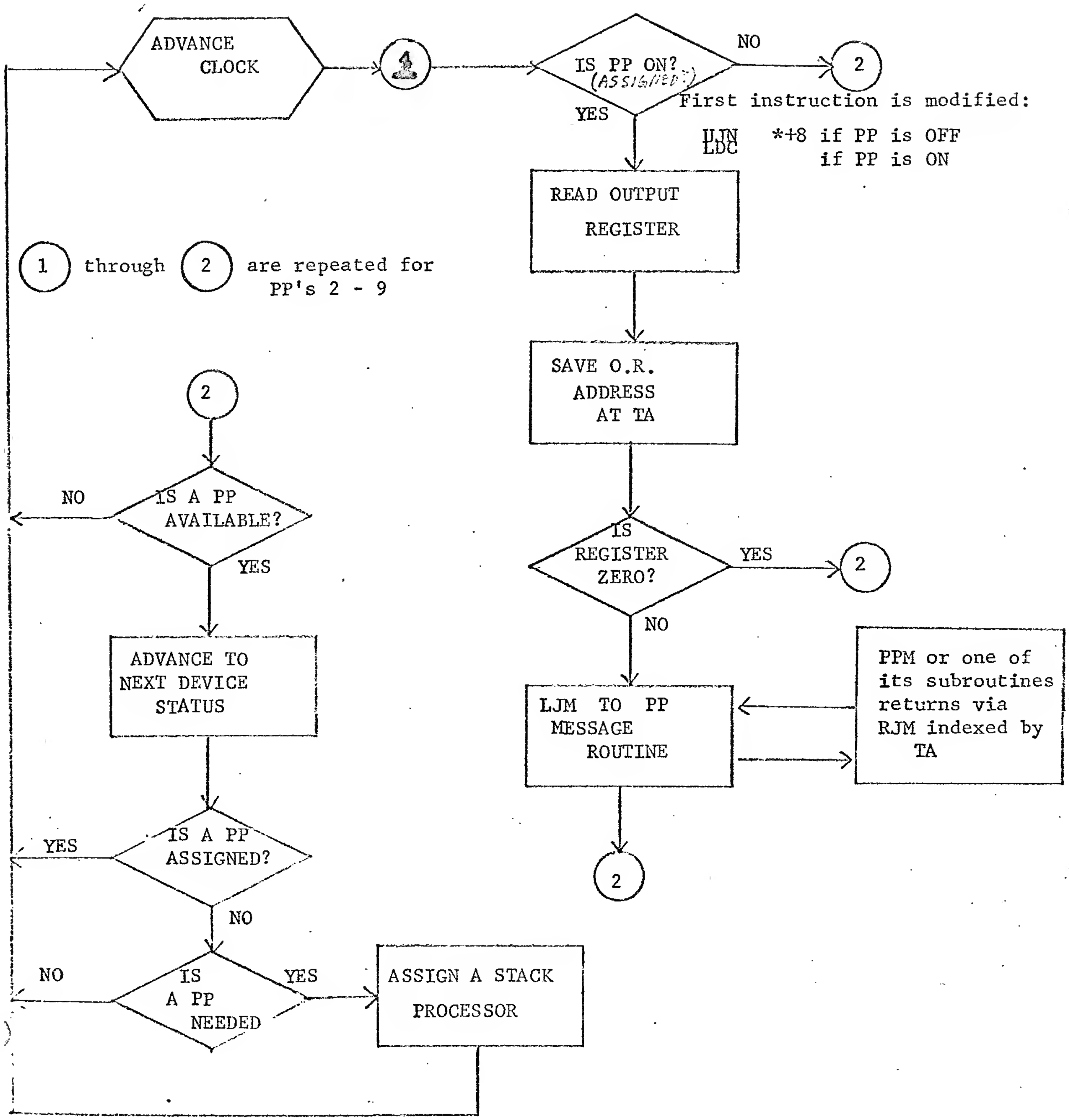


LOOP
MAIN LOOP
PPM
Identify Function Request
Function Subroutines
PFA
PF3
PF2
PF1
PF5
AVC
Advance Clock
SEF
Set Error Flag if P=0
PPC
Process Peripheral Call if RA+1 is not zero
PDS
Process Delay Stack
ACP
Advance Control Point
ATC
Assign CPU Time to Control Point
PRC
Recompute Priority Sublevel

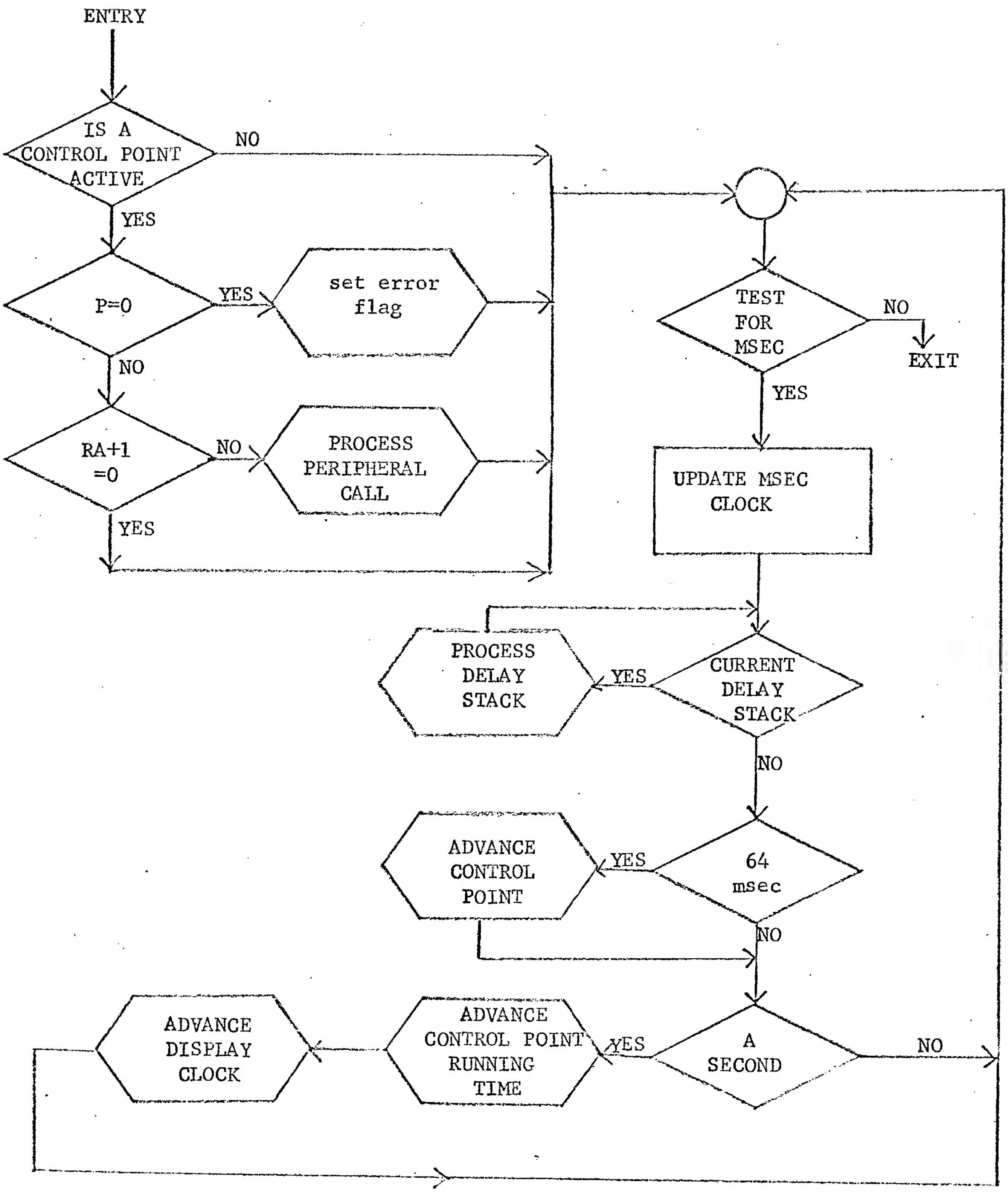
PERIPHERAL PROCESSOR MONITOR FUNCTION



MTR Main Loop



ADVANCE CLOCK



CONCORDANCE OF 2.0/3.0 MONITOR FUNCTIONS

Monitor Function	Mnemonic	Changed	Comments
01	M.DFM	yes	byte 1 = (0 normal request (1 dayfile only ("A" display) - same as 2.0 (2 control point 0 (system) message (4 no "A" display flags may be combined - the upper 6 bits are used by DSD
02	M.RCH	yes	byte 4 = alternate command (may be 0) if channel is assigned byte 4 is set to 7777B hang till assignment achieved by setting byte 4 = 2
03	M.DCH	no	
04	M.PPTIME	no	
05	M.STEP	no	
06	assign track	gone	these functions are no-operations
07	release track	gone	
10	M.RSTOR	yes	also used to request ECS, RBT storage byte 3 = (0 request CM only (1 request ECS only (2 request both (10 request RBT storage (CP 8) (other request CM only for byte 3 = 1 or 2, byte 2 = ECS size in 512 word units for byte 3 = 10, byte 2 = requested low limit for RBT
11	M.CDF	no	now handled by DSD

12	M.DPP	no	MTR will check for auto recall ready
13	M.ABORT	no	
14	M.NTIME	no	
15	M.RCP	no	does not override auto recall
16	M.DCP	no	does not clear auto recall
17	M.PAUSE	no	
20	M.RPP	no	not satisfied until after all RPJs (see 37)
21	M.RCLCP	no	does not override auto recall
22	M.REQP	no	however, response byte moved in Control Point area
23	M.DEQP	no	
24	M.RPRI	no	7777B altered to 7776B
25	M.REM	no	
26	Set dayfile pointers	gone	these functions are no-operations
27	Toggle simulator	gone	
30	M.OPDROP M.SEF	yes	byte 2 = value to set error flag (6 for drop)
31	M.RTAPE	no	
32	M.DTAPE	no	

33	M.AEQP	no	
34	M.EREQS	new	enter words 0 and 1 of message buffer in request stack
35	M.CCPA	new	change control point assignment
			decrement stack count of given Control Point
36	unused		
37	M.RPJ	new	start peripheral job in word 0 of message buffer after given
			msec delay (may be 0)
			reply optional

Subject: SCOPE 3.1 Priority System

The priority of a job in the SCOPE 3.1 system is a twelve bit number. The twelve bits are divided into two fields. The high-order two bits are designated the priority level, the low-order ten bits are the sub-level.

The priority level is obtained from the priority field (P) of the JOB card. The four possible levels and their meaning are:

<u>Level</u>

- 0 Normal Priority
- 1 Normal Expedite (Requires signature of 101 or 595 supervision)
- 2 Special Expedite (Requires signature of Manager)
- 3 Panic Expedite (Requires signature of a Director or higher)

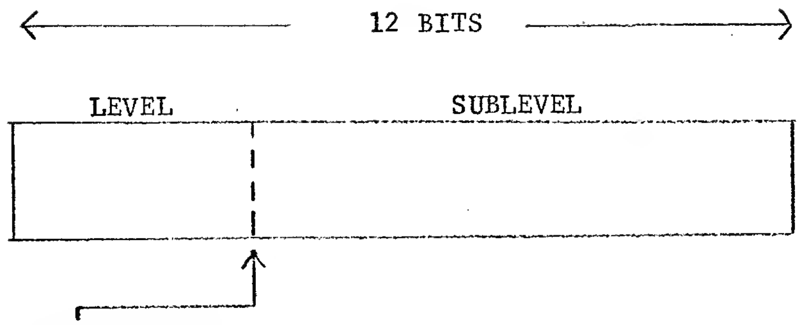
The ten-bit sub-level is initially determined from the time (T) and central memory (CM) estimates. Jobs in the input queue will periodically have their sub-levels increased. Thus the priority increases as a function of time until the job is assigned to a control point. Once a job is at a control point, its sub-level will be re-computed at regular intervals in an attempt to balance the utilization of the CPU between compute-bound and I/O-bound jobs. I/O jobs will tend to have their priorities increased; while the priorities of compute-bound jobs will tend to decrease. The priority sub-levels of jobs in the output queue (PRINT) will be inversely proportional to the amount of output to be printed.

It should be noted that the priority level will never change. That is, a priority 0 will never become a 1 or vice versa.

The following examples demonstrate the initial priority determination. All numbers are expressed in octal.

Job Card Parameters			Priority
P	T	CM	
0	100	60000	1400
0	600	60000	1000 (0000 + 1000)
1	600	60000	3000 (2000 + 1000)
2	600	60000	5000 (4000 + 1000)
0	100	120000	1100
0	600	150000	0440
0	100	40000	1700

PRIORITY



DETERMINED BY:

IP.MPR - THE MAXIMUM PRIORITY LEVEL ASSIGNABLE BY USER.

INSTALLATION OPTIONS AFFECTING PRIORITY (RE) EVALUATION

- IP.LV~~G~~F - LEVEL ABOVE WHICH SUBLEVEL WILL REMAIN FIXED.
- IP.IQD - DETERMINES DELAY BETWEEN INCREMENTING SUB-LEVEL OF A JOB IN THE INPUT QUEUE.
- IP.OQD - DETERMINES DELAY BETWEEN INCREMENTING SUB-LEVEL OF A JOB IN THE OUTPUT QUEUE.
- IP.CPD - DETERMINES DELAY BETWEEN REEVALUATIONS OF SUBLEVEL OF JOBS IN CP.
- IP.OSW - PARAMETER IN DYNAMIC REEVALUATION ALGORITHM FOR WEIGHTING THE FORMER SUBLEVEL.

JOB CARD MACROS

WEIGHT	FIELD, RELATION, VALUE, ADDITIVE
OCTAL	FIELD
DECIMAL	FIELD

FIELD mnemonics:

CM = CENTRAL MEMORY REQUIREMENT

T = OVERALL JOB TIME LIMIT

EC = ECS MEMORY REQUIREMENT

RELATION mnemonics:

GE

LE

VALUE units:

$VALUE_{CM}$ = 64 WORDS

$VALUE_{T}$ = 8 SECONDS

$VALUE_{EC}$ = 512 WORDS

INITIAL SUBLEVEL COMPUTATION EXAMPLES

INSTALLATION SPECIFIES:

```
DECIMAL    T
WEIGHT   T, LE, 8, 10B
WEIGHT   CM, LE, 400B, 4
WEIGHT   T, LE, 4, 20B
WEIGHT   EC, GE, 100B, 100B
```

SAMPLE JOB CARDS:

```
JOB1, T40, CM30000.
   SUBLEVEL = 10B + 4 = 14B
JOB2, T20, CM20000.
   SUBLEVEL = 10B + 4 + 20B = 34B
JOB3, T8000, CM30000, EC200.
   SUBLEVEL = 4 + 100B = 104B
```

DYNAMIC SUBLEVEL REEVALUATION FORMULA

$$S_{i+1} = \left[\left(2^{(IP.CPD+6)} - CPT\right)/ADJ + \left(2^{IP.OSW}-1\right) S_i \right] / 2^{IP.OSW}$$

WHERE:

S = CURRENT PRIORITY SUBLEVEL

CPT = CENTRAL PROCESSOR TIME USED SINCE LAST EVALUATION

ADJ = ADJUSTMENT FACTOR TO NORMALIZE TIME UNITS

$$ADJ = 2^{(IP.CPD+6) - NBS}$$

WHERE:

NBS = THE NUMBER OF BITS IN THE SUBLEVEL.

PRIORITY SUBLEVEL REEVALUATION EXAMPLE

IF:

IP.MPR = 63, THE PRIORITY LEVEL MAY RANGE FROM 1 to 77_8 AND WILL OCCUPY THE HIGH ORDER 6 BITS OF THE PRIORITY ITEM.

IP.LVF = 61, THE UPPER TWO PRIORITY LEVELS (76_8 AND 77_8) WILL BE FIXED, AND NO RECOMPUTATION OF SUBLEVELS WILL TAKE PLACE.

IP.CPD = 4, THE PRIORITY SUBLEVEL WILL BE RECOMPUTED EACH $2^{(IP.CPD+6)}$ MSEC, OR ABOUT ONCE A SECOND.

IP.OSW = 2, THE OLD SUBLEVEL WILL CARRY A WEIGHT OF $2^{(IP.OSW)} - 1$, OR 3.

ADJ WILL HAVE A VALUE OF $2^{(IP.CPD+6) - NBS} = 2^{(4+6-6)} = 16$.

THUS THE REEVALUATION COMPUTATION WOULD BE:

$$S_{i+1} = (1024 \text{ MS} - CPT)/16 + 3S_i/4$$

Assume four jobs running at the same priority. If all four were demanding all of the CP time, each would receive 25% of the CP time. Job A is I/O bound and cannot use more than 10% of the CP time. That leaves 30% for each of the other three jobs. Job B is also I/O bound but can use 40% of the CP time. C and D can make use of all the CP time they can get. The following chart shows how these jobs would share the CP time. Within the chart the first number is the value of the priority sublevel and the second is the percentage of CP time actually used during the period. The example is in decimal for easier comprehension.

PRIORITY / % CP TIME USED

JOB	FIRST PERIOD	SECOND PERIOD	THIRD PERIOD	FOURTH PERIOD
A	50/10	60/0	70/10	75/10
B	50/40	52/0	64/40	63/40
C	50/50	50/0	62/50	59/50
D	50/0	62/100	46/0	59/0

	FIFTH PERIOD	SIXTH PERIOD	SEVENTH PERIOD	EIGHTH PERIOD
A	78/10	81/10	83/10	84/10
B	62/0	71/40	68/40	66/0
C	56/0	67/50	62/0	71/90
D	69/90	54/0	65/50	61/0

	NINTH PERIOD	TENTH PERIOD	ELEVENTH PERIOD	TWELFTH PERIOD
A	85/10	86/10	87/10	87/10
B	74/40	70/40	67/0	75/40
C	55/0	66/50	62/0	71/50
D	70/50	65/0	73/90	54/0

	THIRTEENTH PERIOD	FOURTEENTH PERIOD	FIFTEENTH PERIOD	TOTALS
A	87/10	87/10	87/10	9.3%
B	71/40	68/0	76/40	26.7%
C	65/50	61/0	70/50	32.7%
D	65/0	73/90	54/0	31.3%

STORAGE MOVE EXAMPLE - All figures $/100_8$

Control Point 5 Requests an FL of 300

Control Point	Before RA	Before FL	Before UAS	After RA	After FL	After UAS
0	0	142	0	0	142	0
1	142	33	0	142	33	0
2	175	31	0	175	31	0
3	226	0	500	226	0	0
4	726	20	130	226	20	0
5	1076	100	0	246	300	430
6	1176	150	0	1176	150	0
7	1346	0	430	1346	0	430

SYSTEM DISPLAY & DAYFILE PROCESSING. 6-1+6-2

DSD OPERATOR ENTRY CHANGES

Deletions:

SIM.

DATE.

RECOVER.

n.EXPRESS.

Additions:

n.STEP.	Places control point n in STEP mode. Only one control point may be in n.STEP at any one time. Former STEP entry still valid for system STEP mode.
n.LOADX.	Loads jobs from non-3.0 tapes.
n.BLANK.	Prepares degaused or blank tapes for use by SCOPE 3.0 by writing blank labels (double tape mark) at start of tape.
n.RECHECK.	Rechecks tape labels following pause for operator action due to label error.
n.VRN,xxxxxx.	Enters visual reel number up to 6 digits long following request for same by system.

Modifications:

n.DAYFILE,uu.	Dumps system dayfile to equipment type specified by parameter uu. May be LP, CP, or MT.

Internal Changes:

n.DIS.	May not be entered when system or a control point is in STEP mode.

The following type-ins will cause DSD to be temporarily assigned to the specified control point while the requested operation is being performed.

n.ENDx.

n.GO.

n.OFFSWx.

n.ONSWx.

n.RECHECK.

n.REPEATx.

n.SUPPRESSx.

Display Changes:

The format of the H display has been modified to display only selected files. The format may be altered by typing in H,xy. where x and/or y may be:

I -- INPUT

Ø -- OUTPUT

P -- PUNCH

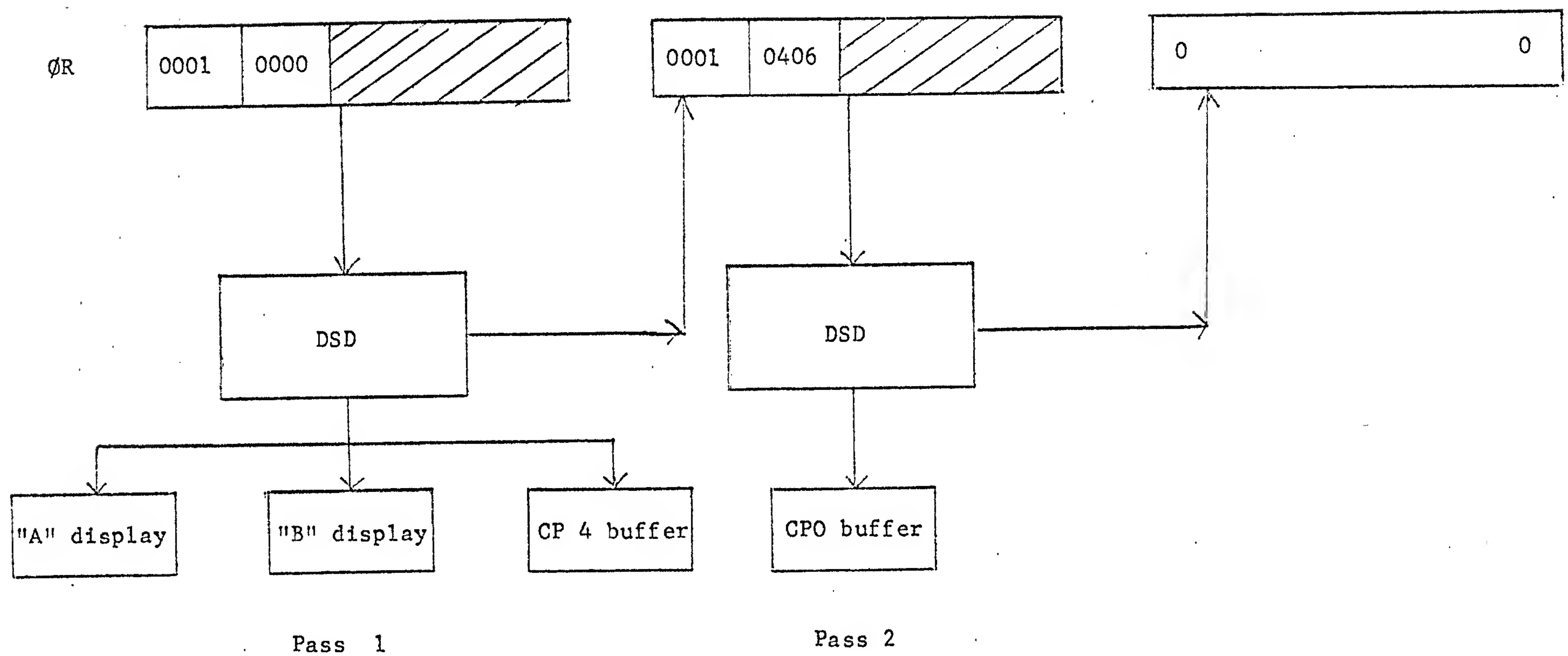
C -- COMMON

L -- LOCKED

FET AT CONTROL POINT 0

DAYFILE		STATUS/CODE
		FIRST
		IN
		OUT
FNT ADDRESS		LIMIT
		MAX

DAYFILE MESSAGE REQUEST PATH (MESSAGE FOR CP 4)



Stack Processor Orders.

Note: These codes are communicated to the stack processor. They are not the codes used in the code/status word.
Order codes requiring a specific request format are indicated. In most cases the format is determined by flag bit settings rather than order code.

O.READ (00): Read into central memory until a short PRU is encountered or the buffer is full(IN=OUT).

O.RDSK (01): Read into central memory until a short PRU is encountered or until the buffer is full. Set the FST to reference the first PRU following the first end-of-record of level x or greater. The level is given in the high-order 6 bits of the order byte. (ie. 1401 would request a read, then would require positioning following an EOR of level 14 or greater)

O.RCMPR (02): Read into central memory after dropping the first three CM words of the first PRU. This is used by STITCH for loading a program from the system library, eliminating the three word header added to system programs by EDITLIB.

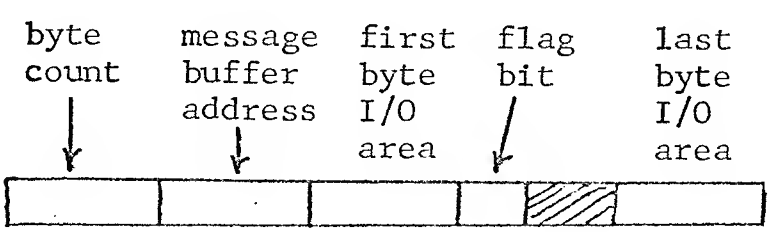
03 : Undefined. Will result in an invalid stack entry message and may abort the control point.

O.WRT (04): Write full PRU's from central memory.

O.WRTR (05): Write from central memory, ending with a short PRU of the level specified in the high-order six bits of the order byte. If an EOF flag bit is found in this order, a zero length PRU of level 17 will be written following the short PRU terminating the record.

06,07 : Undefined.

O.RDP (10) : Read into the requesting PP's memory until a short PRU is encountered, or until the input area is full. For this and other PP I/O orders the format of the second word of the request will be:



O.RDPNP (11): Read into requesting PP after dropping first three cm words of the first PRU. This is used for all PP system program calls.

O.SKP (12): Skip forward n records of level x or greater. The level is specified in the high six bits of the order byte; the number of records to be skipped is given in the third byte of the second word of the order. No data is transmitted

O.SKB (13): Skip backward n records of level x or greater. The level is specified in the high six bits of the order byte; the number of records to be skipped is given in the third byte of the second word of the order. No data is transmitted.

O.WRP (14): Write from requesting PP, full PRU's only.

O.WRPR (15): Write from requesting PP, ending with a short PRU of the level specified in the high order six bits of the order byte. If an EOF flag bit is set in this order, a zero length PRU of level 17 will be written following the short PRU terminating the record.

O.BPRU (16): Backspace n PRU's. The number of PRU's to be backspaced is given in the third byte of the second word of the order. Note that this requests repositioning defined by physical rather than logical units. No data is transmitted.

O.RCHN (17): Release chain. All record blocks assigned to a file are released, and the RBT word pairs containing them are released. The FST is reset to an empty condition, if its address is supplied in the order.
Note: requests 16 and 17 requeire no communication with the device, and therefore are given the highest priority in the search for the next order to be executed. All other requests are assigned priority based on repositioning required.

REQUEST FOR PP TRANSACTION

<table>
<tr><td>WORD 1
IF FNT bit = 0</td><td colspan="3">Address of FNT word</td><td></td><td colspan="7" rowspan="2">Order
(specifies PP
transaction)</td><td rowspan="2">Control
Point
Number</td><td rowspan="2">Physical
Unit</td></tr>
<tr><td>WORD 1
IF FNT bit = 1
(NO FNT)</td><td>RBT word</td><td>RBT ordinal</td><td>byte</td><td>PRU</td></tr>
<tr><td>WORD 2</td><td>PP word count</td><td colspan="2">Address of
PP message word</td><td>PP FWA</td><td>RECALL</td><td></td><td>FNT</td><td>FET</td><td></td><td>AVAILABLE</td><td></td><td colspan="2">PP LWA + 1</td></tr>
</table>

Release EOF
or
After EOR

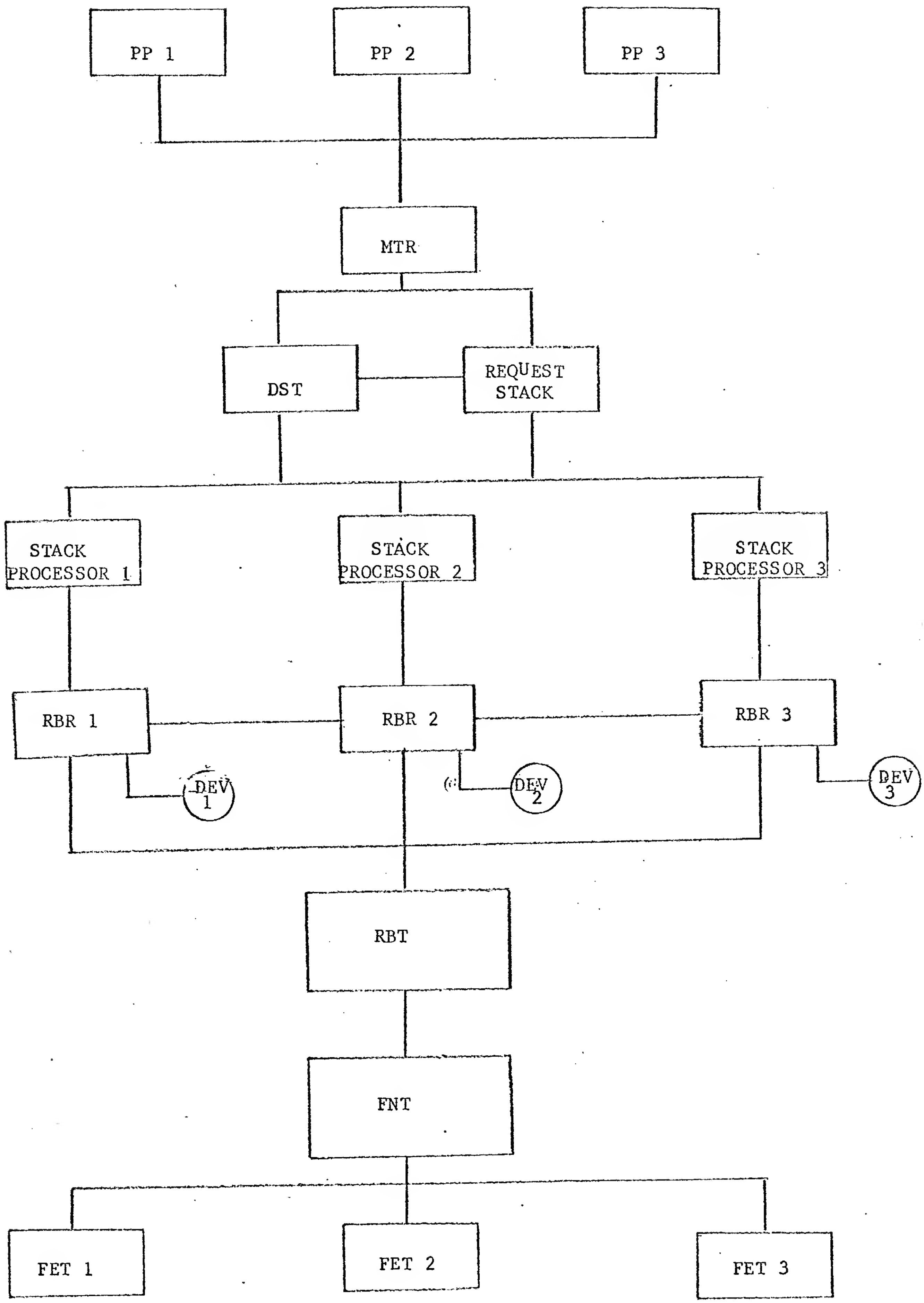
REQUEST FOR CM Transaction

<table>
<tr><td>Word 1
if FNT bit=0</td><td colspan="3">Address of FNT word</td><td></td><td rowspan="2"><u>Order</u>
(specifies CM
transaction)</td><td rowspan="2">Control
Point
Number</td><td rowspan="2">Physical
Unit</td></tr>
<tr><td>Word 1
if FNT bit=1
(no FNT)</td><td>RBT word</td><td>RBT ordinal</td><td>byte</td><td>PRU</td></tr>
</table>

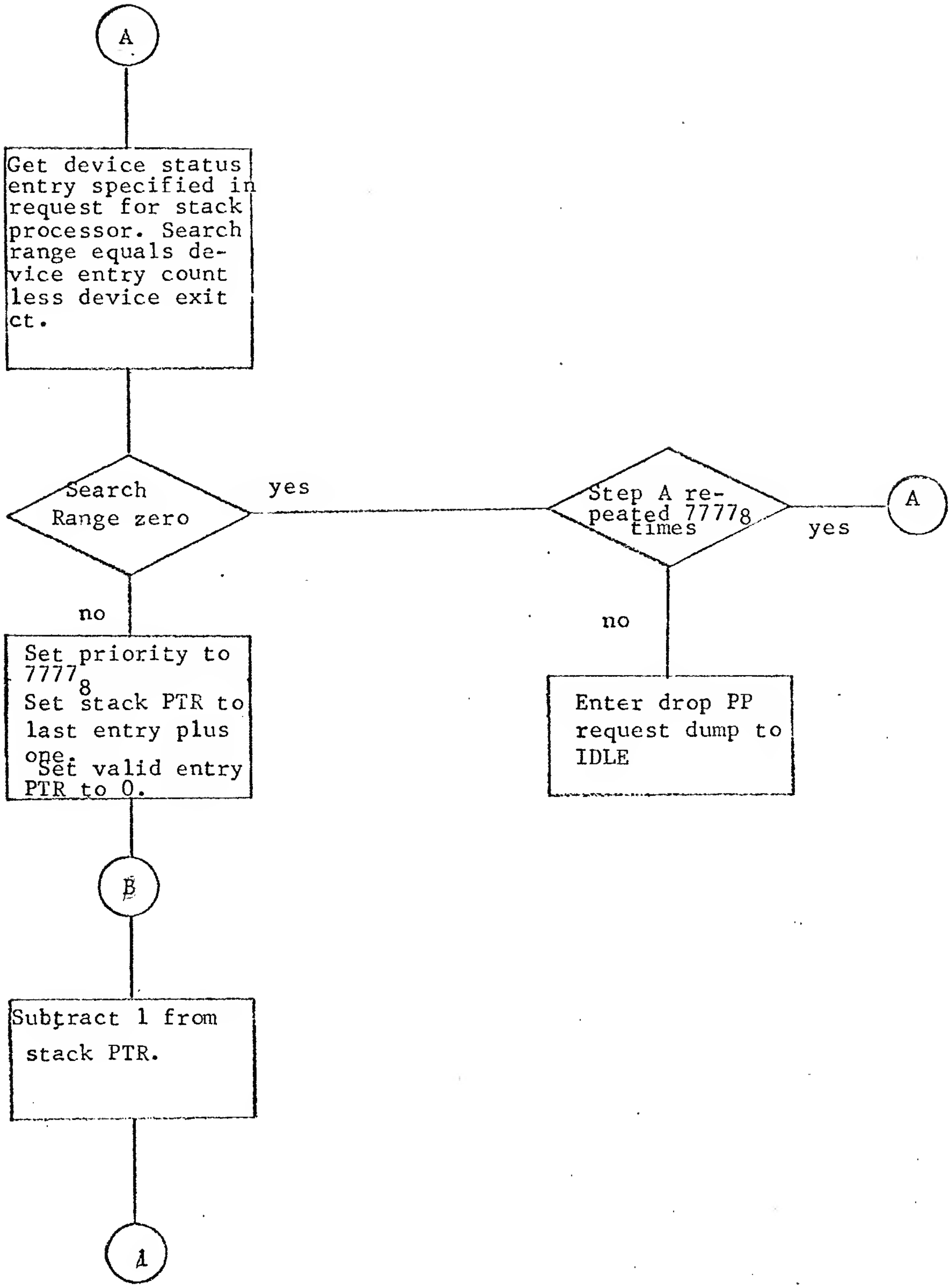
<table>
<tr><td>Word 2</td><td>FET FWA</td><td>FIRST for transmission
or
RECORD (PRU) COUNT-n</td><td>R
E
C
A
L
L</td><td></td><td>F
N
T</td><td>F
E
T</td><td></td><td>A
V
A
I
L
A
B
L
E</td><td>LAST
for transmission</td></tr>
</table>

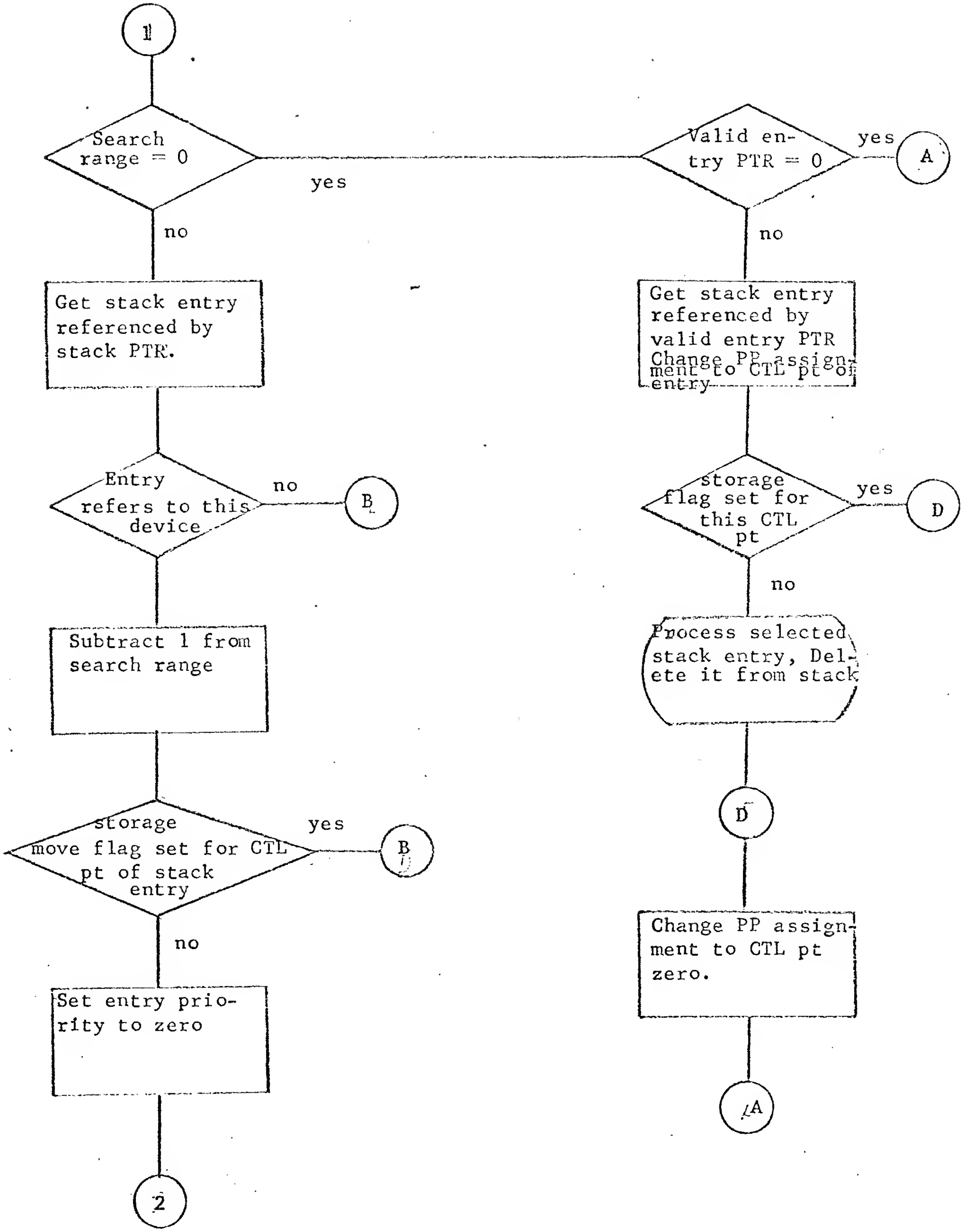
Release EOF
or
After EOR

7-5

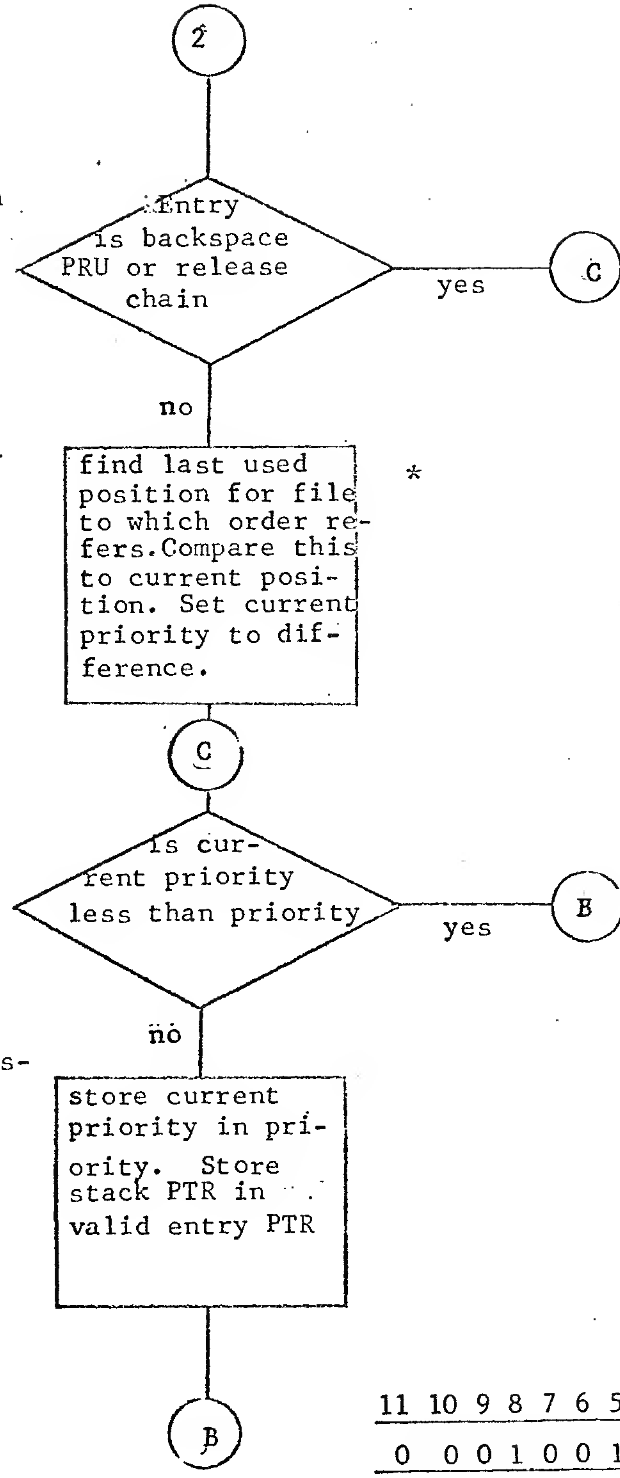

PP 1
PP 2
PP 3
MTR
DST
REQUEST STACK
STACK PROCESSOR 1
STACK PROCESSOR 2
STACK PROCESSOR 3
RBR 1
RBR 2
RBR 3
DEV 1
DEV 2
DEV 3
RBT
FNT
FET 1
FET 2
FET 3

DISK STACK PROCESSING




1
Search range = 0
yes
no
Get stack entry referenced by stack PTR.
Entry refers to this device
no
B
Subtract 1 from search range
storage move flag set for CTL pt of stack entry
yes
B
no
Set entry priority to zero
2
Valid entry PTR = 0
yes
A
no
Get stack entry referenced by valid entry PTR Change PP assignment to CTL pt of entry
storage flag set for this CTL pt
yes
D
no
Process selected stack entry, Delete it from stack
D
Change PP assignment to CTL pt zero.
A

Note: Backspace PRU and release chain orders do not reference disk and therefore require no positioning.



* Current position and last-used position are both used in the format shown below (Fig A). By use of logical minus, and subsequent editing of the result, repositioning is expressed as a binary number. (see example, Fig. B)

Track							Zone GP	Zone		Half track	RB Flag
11	10	9	8	7	6	5	4	3	2	1	0

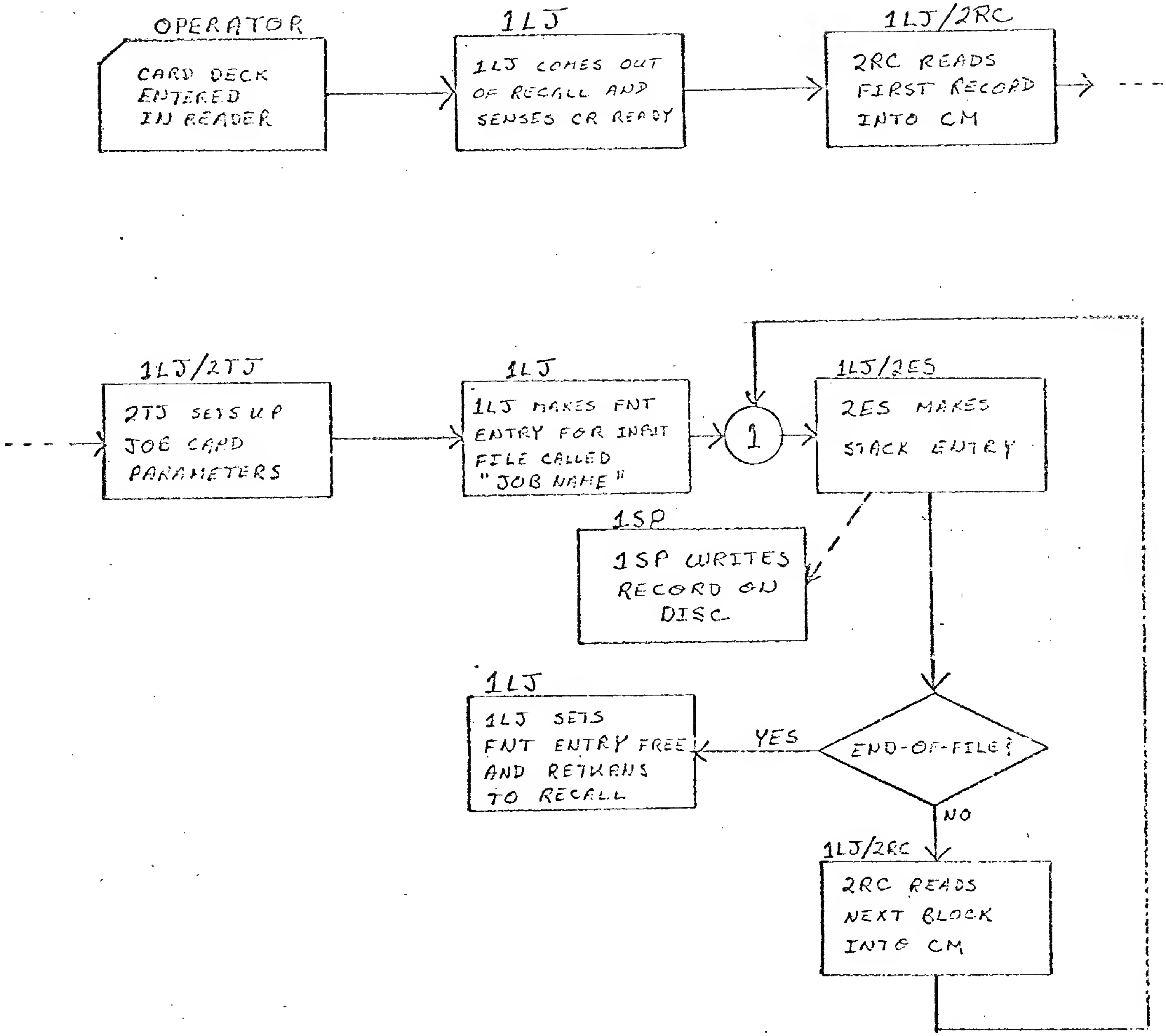
Fig. A

11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	1	0	0	1	0	1	1	0	1	current position
0	0	0	0	1	0	1	1	1	0	1	1	last used file pos.
0	0	0	1	1	0	0	1	0	1	1	0	logical difference
0	0	0	1	1	0	0						edited current priority

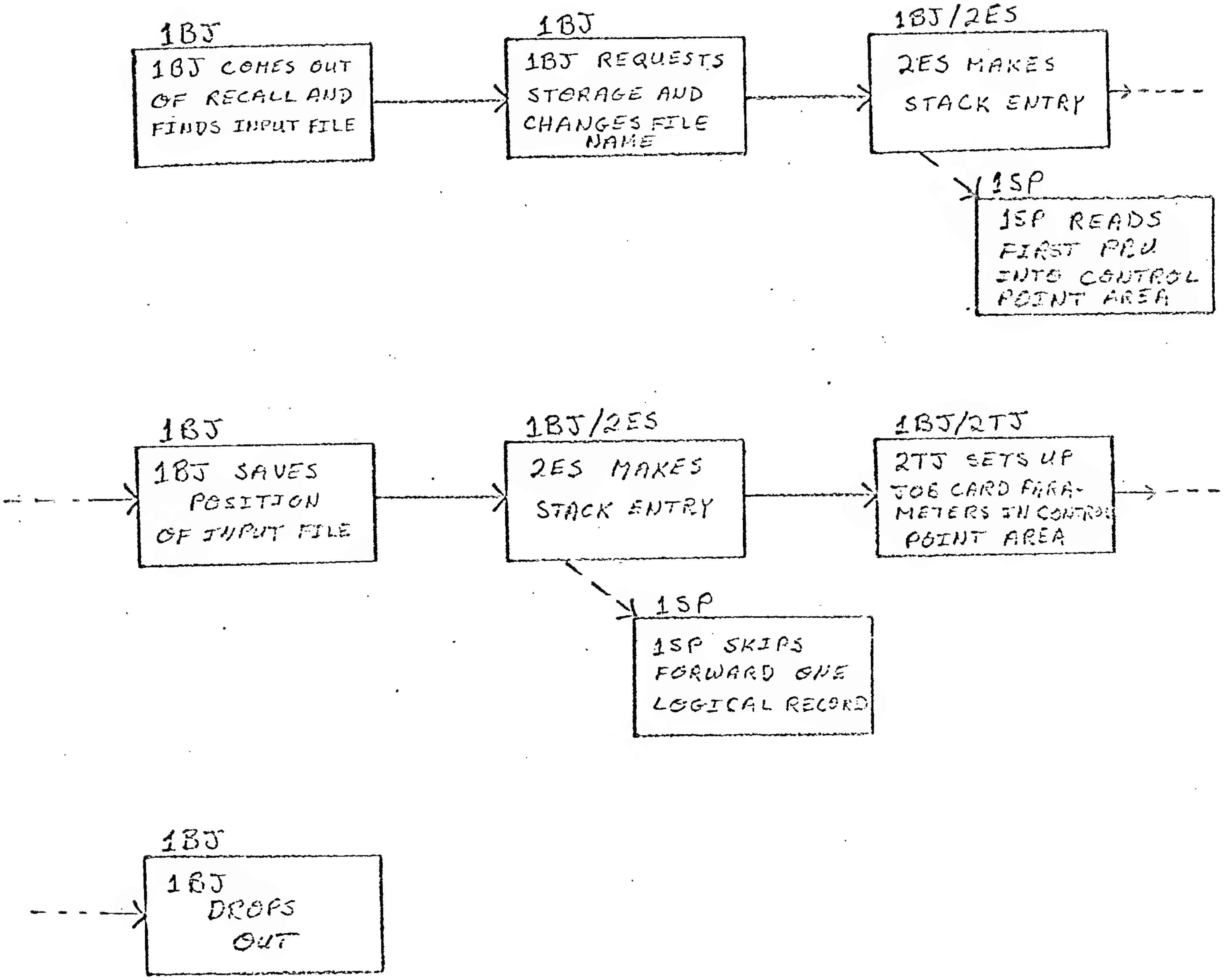
Fig. B

JOB FLOW

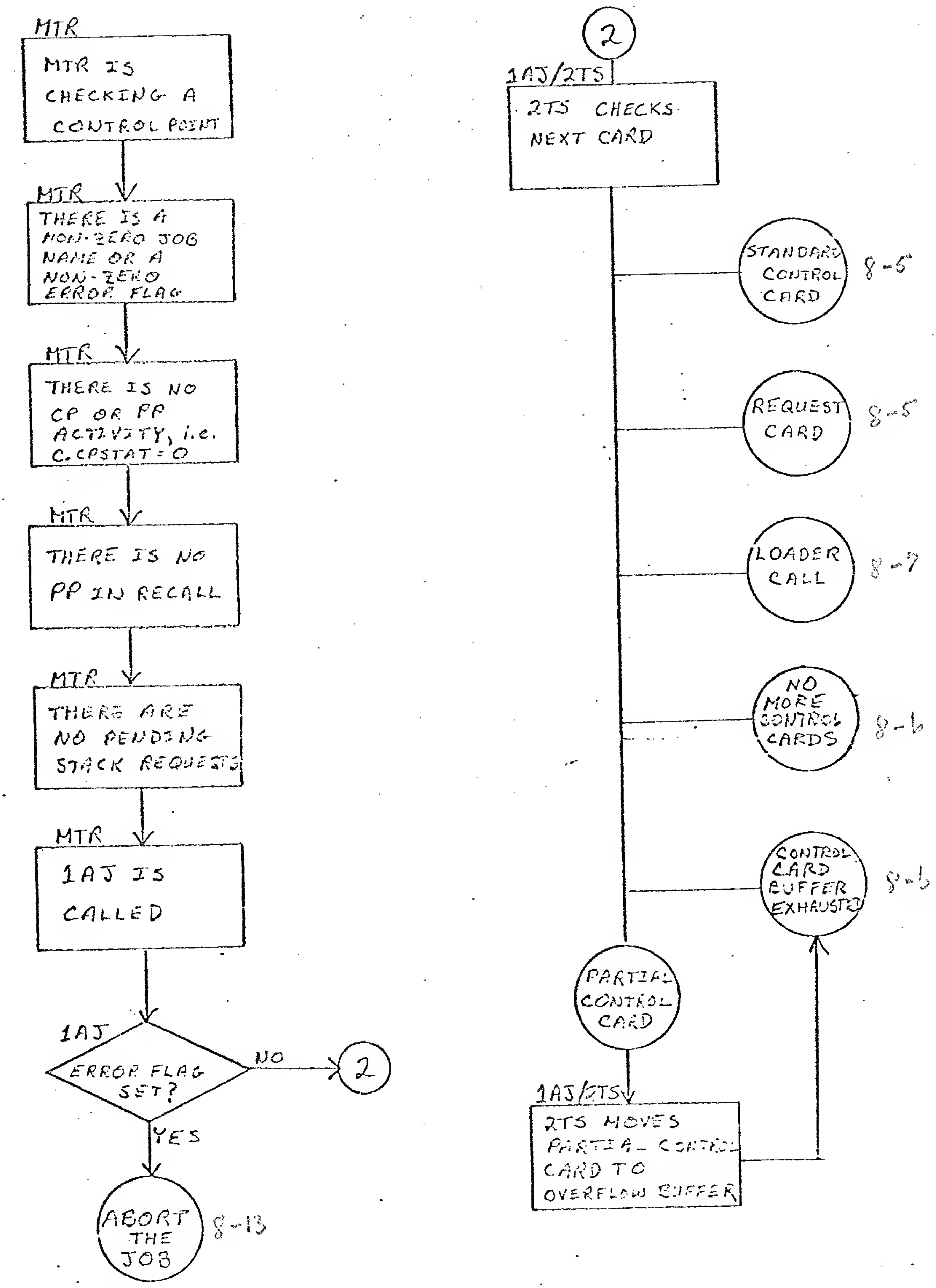
LOAD JOB



BEGINNING A JOB



PROCESSING A JOB
(ADVANCE JOB)



The following paper will describe the major SCOPE 3.0 Peripheral Processor routines involved in getting jobs into, through, and out of a 6000 series computer. Familiarity with the hardware and with the external characteristics of SCOPE 3.0 is assumed.

The discussions will begin with the system as it stands immediately after completion of a dead-start load. The system monitor (MTR) is in PP0, idling as it waits for a request to process. (It should be realised that MTR will be continually searching for requests while other processes are being performed elsewhere. The actual functions done by MTR will be introduced and described as needed). The system display package is in PP9, initially displaying on the left console the message :

ENTER DATE

MTR &DSD are permanently assigned to PPs 0, and 9, respectively.

Central memory contains the CM resident tables, flags and routines, collectively known as CMR. The major portions of CMR reside in low numbered central memory, although there are some dynamically variable disc tables located in high core. Again relevant tables and flags will be introduced and defined as needed.

The last component of an idling SCOPE 3.0 system is the Peripheral Processor resident program. A copy of PP resident is loaded into each of the pool PPs (PPs 1 through 8). PP resident contains a number of small service routines which are commonly needed by most PP overlays; additionally, PP resident contains an idle loop which is executed initially and whenever the pool PP is not assigned to a specific task. Each of the pool PPs is aware of its own number and each PP has assigned to it a unique 8 word area in CMR known as a PP communication area. Each communication area has a location called a PP input register, one called a PP output register, and six called the PP message buffer. The PP resident idle loop consists primarily of continual examinations of the PP input register; a zero value input register is the idling condition. A non-zero value indicates that this PP is being requested to perform a task and therefore must load an overlay.

PP overlays are listed in the system library directory which is a part of CMR; the location of the overlay is also given. An overlay may be located either on disc or in central memory; the overlays which are frequently needed are usually kept in central memory, since loading speed is maximized. There are two basic types of PP overlays: primary and secondary. Primary overlays are those programs which may be requested through a PP input register; they are loaded by PP resident at 773B (there is a non-executable 5 PP word header appended by COMPASS) and entered by a

LJM 1000B.

Secondary overlays are essentially subroutines of one or more primary overlays; they are loaded into higher PP core when the controlling primary overlay makes a request to a PP resident routine (R.ØVL) to do so. The majority of secondary overlays are coded to be loaded at 1773B and entered by a

RJM 2001B.

However, there are several which must be loaded at 2773B, 3773B, etc. Also, there are a few secondary overlays which utilize a set of macros (called RELØC) which have the effect of making the overlay self-relocating; such overlays may be loaded anywhere, providing of course that they do not wrap around the end of memory. This latter condition is not detected by the system; it will eventually be recognised by the operator since the destruction of a PP will (usually) destroy the system ultimately.

All PP overlays have three-character names; there are certain conventions which are followed with respect to the first character when assigning names. Primary overlay names either begin with an alphabetic character or the digit one (1). Alphabetic overlays may be called from a running program, a control card (in some cases), or from within the system; numeric overlays may only be called from within the system, i.e. from a PP. Secondary overlay names usually begin with the digit which is closest to the 1000_8 word block at which the overlay is to be loaded; e.g. 2TR is loaded at 1773B. No conventions have been established for the second and third characters, except that they should have some (be it ever so tenuous) mnenonic value.

Now, to return to the idling system, let us begin to do some processing. To set

the system going the operator must type in today's date in the correct format. Until this is done, DSD remains in an idle loop waiting for keyboard input. When the date is received, its legality is checked and DSD enters the value into a flag location in CMR for use by the remainder of the system. DSD then brings up the A (dayfile) display on the left console and the B (control point) display on the right console.

At this point, normally the operator would type in AUTØ to prepare the system for full multiprogramming; for instructional purposes, we shall simplify the procedure and only type in

EAD
ACKAGE

1. READ.

DSD has been cycling through its main loop which consists primarily of refreshing the console displays and checking for keyboard input. When the above command is received, DSD branches to the routine which processes a request for the READ package. It is known to DSD that the name of the PP overlay comprising the READ package is 1LJ (load job) and that 1LJ must be assigned to a pool PP.

1. DSD posts in its output register a request to MTR (DSD also has a PP communication area. Its input register is not used since PP9 is not a pool PP and cannot be assigned tasks by MTR). This particular request will be a "request peripheral processor" (M.RPP). (It will be seen later that M.RPP is a subset of another monitor function and exists only for historical reasons.) DSD's output register will contain the code for M.RPP and the control point number to which the pool PP will be assigned, in this case, 1. As is usual, additional information is given in the PP message buffer; in the case of an M.RPP function, the first word of the message buffer is expected to contain the input register setting for the requested pool PP.

2. Part of MTR's loop consists of checking the output registers of the 8 pool PPs and DSD. A Zero value constitutes no request; a non-zero value indicates some function to be performed. In this case, MTR will pick up the first word of DSD's message buffer, insert the control point number (1), and place the word into the input register of an available PP. In this particular situation, the PP chosen will be PP1 since nothing

else is happening; however, in other cases, any of the other 7 pool PPs might have been selected.

3. The PP resident in PP1 will now notice that its input register is non-zero, specifically that the high order 18 bits contain the display code name of the PP primary overlay which is required to perform the desired task, (1LJ), the next 3 bits are zero, and the next 3 bits contain the control point to which the task is assigned (1). PP resident looks up the name 1LJ in the system library directory, reads the overlay into its own memory and transfers control to 1LJ at 1000_8.

4. 1LJ first performs the initiallization which is common to nearly all primary overlays. The contents of the PP input register are read into PP low core; the control point number is used to compute the address of the control point area. (CMR contains a control point area for each of the seven control points. This area contains information about the state of the job (if any) currently assigned to the control point. Control point area contents will be described as needed during the course of this discussion.) The control point reference address (RA) and field length (FL) are obtained from the control point area. These quantities are retained in PP low core for use by the primary overlay and also any secondary overlays called during the performance of the assigned task.

5. 1LJ enters the job name READ into control point area 1 in order that DSD may display, for the operator's information, the current job. Prior to this time, control point 1 (and all other control points) had a zero job name indicating that the control point was vacant.

6. 1LJ then posts a request (M.RSTØR) in its output register for central memory storage for buffer space. Again, MTR will sence the request and in this case will immediately give control point 1 the necessary storage. Granting this request consists only of setting the desired FL into control point area 1 and resetting the RAs of control points 2 - 7 equal to RA+FL of control point 1.

7. At this point, we shall introduce a table in CMR called the Equipment Status Table

(EST). The EST contains an entry for each I/Ø device which is present in the configuration; one word is required for each device. Each entry contains a 2-character display code device type, a flag field to indicate assigned or available, and hardware information (channel number, equipment number etc.) Entries may be ordered whimsically, but the relative position (EST ordinal) with respect to the first word of the EST must be known by the operator in order to assign equipment.

Returning to the job processing, 1LJ now posts a request (M.REQP) in its output register for equipment of type CR (card reader). MTR searches the EST for such a device and will select the first unassigned card reader (lowest numbered ordinal) that it finds. Hence, the order of EST entries must also be known to determine which card reader goes to which READ package. (Note that if a second READ package were assigned to another control point, it would handle the card reader with the next highest EST ordinal, and so on.) Each copy of 1LJ can handle only one card reader. MTR marks the selected card reader EST entry as assigned and returns the EST Ordinal to 1LJ via 1LJ's message buffer.

8. 1LJ then sets up the appropriate hardware instructions to sense the status of its assigned card reader and let's assume first that it is not ready, i.e. the operator has not yet put any job decks in the hopper. The system, in general, tries to keep as many pool PPs available as possible and since it may be a while before the card reader is readies, we introduce a new concept, PP recall. 1LJ uses the old-fashioned form of PP recall for historical reasons only. Old-fashioned PP recall required the use of a location in the control point area, known as the PP recall register. 1LJ writes the contents of its input register into the PP recall register and then issues a monitor request (M. DPP) to return the PP to the pool. MTR examines control point PP recall registeres during its main loop and if one contains a non-zero quantity, the register contents are placed in the input register of an available PP and the recall register is reset to zero.

Modern PP recall requires the use of a monitor function (M.RPJ - request peripheral

job) followed by an M.DPP. M.RPJ allows the specification of a time interval which is to elapse before a pool PP is assigned (the time interval may be zero, meaning that assignment should be made as soon as possible). It should be noted that:

(a) M.RPJ can (and should) replace M.RPP, and

(b) M.RPJ would allow two or more PPs at a single control point to enter recall simultaneously; use of the PP recall register does not permit this ability.

MTR maintains a list called the PP delay stack in PP0 memory of PP programs to be called after a specified interval; the delay stack is examined each time the clock is updated. It can be seen that 1LJ could make effective use of the delay stack feature since the event it is awaiting (card reader ready) is measurable in human time; a pool PP would be required less frequently using M.RPJ than by using the PP recall register. But I digress. 1LJ is now said to be in recall; when it is reloaded, it starts anew, i.e., returns to step 4. This is necessary since it cannot rely on being loaded into a properly initiallized PP. However, 1LJ need not repeat all the steps since it can determine that control point 1 already has CM storage and a card reader. Again it tests the status of the card reader, returning to a recall state if it is not ready. This time, we shall assume a ready condition.

9. Other than the test for card reader ready, 1LJ has no I/O driver capability; it must load a secondary overlay. The card reader driver overlay is named 2RC. 2RC is loaded (as are all other secondary overlays) by presetting 2 locations in PP low core with the display code overlay name, entering the load address in the A-register, and executing a RJM to the PP resident routine, R.ØVL. 1LJ then enters 2RC which reads cards until either the central memory buffer is filled or until an end-of-record or end-of-file card is encountered, whichever occurs first. When 2RC has finished processing, it returns to 1LJ.

10. 1LJ then loads 2TJ (translate job card). 2TJ will locate the first card image in the buffer and check its validity as a job card. (You will recall that the first card of the first logical record of a job is required to be a job card.) Additionally, 2TJ will save in PP low core the job name (appending a job count to keep the name unique),

the required field length and priority for later use. Control is returned to 1LJ.

11. Assuming that the job card was indeed valid, 1LJ now loads 2BP (check buffer parameters). 2BP is responsible for checking that various file tables (which are described elsewhere) contain legal values and for setting up PP low core locations in the form required by hardware drivers to be loaded subsequently.(Note: logically, 2BP should have been loaded before 2RC in step 9 above; the reason for not doing so are not known to the author) 2BP is genrally called immediately prior to the loading of any I/Ø driver. It is more important that it be loaded when user I/Ø is involved, however, since one assumes that system routines such as 1LJ are not going to produce erroneous file tables.

12. 1LJ next loads 2ES (enter stack). 2ES is a pseudo I/Ø driver, in a sense. It is loaded and treated as though it actually could read and write an allocatable device (i.e. disc), but it can't. What it does do is to gather together information about the file to be read or written and prepare an entry for a CMR table called the request stack. The necessary information about physical device, file position and so forth is formatted within PP memory and 2ES enters a PP resident subroutine, R.EREQS. R.EREQS in turn searches the request stack for an available location and formats a request to MTR (M.EREQS) to make the entry. (PP resident performs the stack search in order to minimize the monitor loop time; MTR, however, must actually make the entry in order to provide an interlock on the request stack. If two copies of PP resident have simultaneously located the same empty entry, MTR refuses the second M.EREQS function and the rejected PP resident must re-search the stack). When the stack entry has been made, R.EREQS returns control to 2ES which then exits to 1LJ. 1LJ remains in a tight loop, waiting for the central memory buffer to be written to the disc. (The detailed mechanics of disc reading and writing will be covered elsewhere).

13. When the buffer is empty (i.e. the disc operation is complete), 1LJ loads 2BP and then 2RC to refill the buffer from the card reader. Steps 11 through 13 are now repeated

until an end-of-file card has been read and written. It can be seen that there is now an exact logical record-by-logical record copy of an input file on the disc. The only thing that now remains is to dissociate this file from the READ package and make it available for running. This is done by making an entry in another CMR table called the File Name Table or FNT. (The FNT was used during the course of constructing the input file, but in an uninteresting and difficult-to-describe manner, so we shall introduce it here). The FNT contains a 3-word entry for every file known to the system; an entry contains the file name and type, the type of device on which the file resides, location information peculiar to the particular device, current state (busy or not busy), the priority and the control point to which the file belongs. In the case of the input file which we have just constructed, 1LJ will make an FNT entry containing the following items:

- file name; this will be the job name extracted from the job card; this name was made unique by virtue of the fact that 2TJ appended a job count;
- file type: this will be type input. (Other file types will be introduced as needed). All files in the FNT of type input constitute the input queue.
- device type: this will be disc, either 6603 or 6638.
- location information: during the course of constructing the input file, various disc tables were built in central memeory; the FNT entry contains pointers to these tables.
- current state: this will be not busy
- priority: this will be the priority extracted from the job card as the high order part (called priority level) and zero as the low order part (called sublevel). The use of priority bits is described elsewhere.
- control point assignment: this will be zero, indicating that the file is not assigned to any control point.
- additionally, in the case of a file of type input, the FNT contains the central memory and ECS field lengths requested on the job card.

Having entered the file in the input queue (i.e. made the FNT entry), 1LJ returns to check the status of the card reader (step 8), entering PP recall if there are no more cards in the hopper or constructing another input file if there are.

Let's assume there are 5 more jobs, so 1LJ reiterates the above process 3 more times ultimately reaching the state where it is going in and out of recall. We now have an input queue containing 4 jobs (files) it will be necessary to bring up a package to select one or more jobs for execution. Before we do that, however, to keep life simple and for instructional purposes, let us drop the READ package. This is sometimes done during normal running when there are no more job decks to be entered in order to free up the buffer space that 1LJ has.

ROPPING
HE
EAD
ACKAGE

1. The operator must initiate the dropping of any control point; this is done by typing x.DRØP where x is the control point number. In this case, he types:

1.DRØP

Again, DSD will detect keyboard input during its main loop and will branch to process the DRØP command. Processing consists of sending a request (M.ØPDRØP) via DSD's output regist to MTR. The M.ØPDRØP also contains a control point number, in this case, 1. The effect of this function is to cause the monitor to set an error flag within the control point area of control point one.

2. Now you will recall that 1LJ is still popping in and out, testing card reader ready. In addition to this already mentioned test, 1LJ also checks the control point 1 error flag. (It also checks the error flag during processing, so that READ may be dropped in the middle of card reading, but this is not too relevant to the current discussion). If at any time 1LJ discovers that the error flag is non-zero, it will drop its PP (via the M.DPP function) <u>without</u> setting the PP recall register. This leaves control point 1 still occupied but inactive, since there is now no way for any more activity to be performed.

3. Meanwhile, MTR has been cycling through its main loop, part of which consists of an Advance Control Point routine. The Advance Control Point routine is entered once every 64 milliseconds and each time it checks the condition of one control point, beginning with 1 cycling up to 7 and returning to 1. A control point is said to require advancing if all of the following conditions are true:

- There is a job name. (This condition is met since the job name READ has not been cleared.
- There is no central processor activity. (The READ package has never initiated any CP activity)
- There is no peripheral processor activity. (1LJ has dropped itself without recall and it was the only PP which was attached to control point 1).
- There are no PP recall requests (either in the delay stack or in the recall register). (1LJ did not set the recall register before it dropped).
- There are no pending entries in the request stack. (The only request stack entries made by 1LJ are during the course of preparing a job for entry into the input queue and there are no more jobs to be prepared).

It can be seen, therefore, that control point 1 must be advanced.

Advancing a control point, from MTR's point of view, consists of calling 1AJ (advance job) into a pool PP and attaching the PP to control point 1.

4. As will be seen later, 1AJ can follow several paths varying in complexity, depending on the state of the control point which is to be advanced. The case currently under investigation, namely that of dropping a system control point (READ) is probably the simplest and involves the following steps.

1AJ checks the error flag which is set. 1AJ then checks the priority of the control point which is given in the control point area; now, since READ is a system control, it has zero priority. This fact greatly simplifies the process as will be seen later. 1AJ requests the loading of the overlay 2EF (error flag). 2EF checks the actual value of the error flag to determine which message to enter into the dayfile. In this case, the value is 6 (F.ERØD) which directs 2EF to write the following message:

hh.mm.ss	READ	ØPERATØR DRØP
(Time of Day)	(job name)	(message)

2EF returns to 1AJ.

5. 1AJ begins a search of the FNT, looking for (in this case) files which are assigned to control point 1. Each time such a file is located, 1AJ will load the overlay 2DF (drop files). 2DF "removes" the file from the FNT; what constitutes removal of a file is

dependent upon the file type.

It should be specially noted that those files which were entered into the input queue were assigned to control point 0 and are therefore not dependent on the existence of their creating READ package). When all files have been removed from the FNT, 1AJ then searches the EST for equipment assigned to control point 1; in this case it will find exactly one equipment, the card reader. Again, 2DF is called to return the card READER to the pool of available equipment.

1AJ then posts an M.RSTOR function to MTR to request that all central memory storage associated with control point 1 be dropped. Control point area 1 is then cleared, i.e. the job name, etc, are set to zero.

We are now faced with a system in an essentially idle condition except that there are four jobs in the input queue. We must now initiate processing. To do so, we will bring up one copy of the next package at, say, control point 3.

XT 1. The procedure for bringing up NEXT is nearly identical with that for bringing up READ and so we shall abbreviate the description. The operator initiates the process by typing 3.NEXT. It is known to DSD that the name of the PP overlay comprising the NEXT package is 1BJ (begin job). DSD requests, again with the M.RPP function, that 1BJ be assigned to a pool PP at control point 3. MTR will select a pool PP and PP resident will load the overlay 1BJ.

2. 1BJ will perform the usual initialization of PP low core (See READ PACKAGE - item 4) and then enters the job name NEXT into the control point area for display purposes.

3. MTR maintains in CMR a location (T.UAS) which contains the current amount of available central memory and ECS. 1BJ reads these quantities into PP low core and then requests the FNT channel. (A slight digression: MTR maintains a channel status table which includes entries for the 12 hardware channels, and several pseudo-channels.

Before a PP can use one of the hardware channels or alter a table represented by a pseudo-channel, it must request the use of the channel from MTR. This is another method of providing a table interlock.) In the case under consideration, the FNT

channel really need not be obtained since there is no possible conflict; but a more normal situation would include several copies of 1BJ making identical searches and it is considered desirable to execute each job only once.

4. 1BJ performs a search of the File Name Table, looking first for a file of type input. Having located such a file, it compares the requested CM and ECS field lengths with the currently available storage. If there is adequate storage for the job, then the job priority is compared with the priorities of all other jobs which meet the storage requirements. The job with the highest priority will be selected for execution. Also, during the course of every nth FNT search by 1BJ (where n is an installation parameter), the priority sublevel of all files in the input queue will be incremented by one; the sublevel will not be allowed to overflow into the level, however. Hence, it can be seen (assuming that enough storage is always available) that priority level as declared on the job card is the major determinant of the order of execution and that, within a set of jobs of the same priority level, the length of time the job has been in the job queue will determine the order of execution.
If 1BJ had failed to find a candidate for execution, it would now enter PP recall in the old-fashioned manner, i.e. by setting the PP recall register and dropping out. However 4 jobs are known to be in the input queue and so we carry on.

5. The selected FNT entry must now be altered. The file name is changed from the unique job name to INPUT; the type is changed from input to local; and the control point assignment is set to 3. Also, the fields which contained the ECS and CM field length requirements are altered to the more usual disc position information. The position of the input file, at this point, is logically rewound, i.e. the next record to be read is the first (control card) record.

All this means that the user's input file will be available to him alone (since it's assigned to what will soon be his control point) and that it may be referenced simply by calling for the file named INPUT.

6. 1BJ then sets up control point area 3 for the selected job. The job name is

changed from NEXT to the name of the job to be executed. The priority (with the sublevel field reset to zero) is entered into the control point area with the aid of a monitor function (M.RPRI). The current position of the input file is also saved in the control point area for reasons which may become clear later. 1BJ then calls 2ES to format a disc request to skip forward one logical record on the input file.
(You will recall that this first record contains only control cards and is presumably not of interest to the user's program). 2ES, through PP resident and the monitor, enters the request stack, leaves the INPUT FNT entry busy and then returns to 1BJ. 1BJ loops until the FNT entry becomes not busy (request completed). The INPUT FNT entry is now positioned at the user's first data record.

7. 1BJ now indulges in a game of fake-out-the-system, the object of which is to read the first sector of control cards into a small buffer in the control point area. The method used is, briefly, as follows. Temporarily, the FNT entry for INPUT is set to reflect a rewound condition (this can be done since the initial file position was saved in the control point area). 1BJ then sets up its own PP to look as if it belonged to control point 0 rather than control point 3. (Control point 0 is a convenient fiction invented for just such cases as this. The area occupied by CMR is said to belong to control point 0). This is necessary because it is illegal for a PP attached to one control point to request that data be read into the area belonging to another control point. 1BJ then loads 2ES to format a request, in the usual manner, to read data into the sector-sized buffer in the control point area. 1BJ then awaits completion of the operation and restores the proper values in its PP and in the FNT. However, instead of returning the rewound position of the INPUT file to the control point area, it sets the position of the next sector of control cards; this allows 1BJ to perform the above sneaky operation each time another sector of control cards is required.

8. 1BJ next calls 2TJ to reprocess and skip over the job card which is the first card image in the control card buffer. 2TJ will again check the validity of the job card. (Because 1LJ has no facilities for writing an error message for the user, a job with an erroneous job card will be passed along into the input queue so that 1BJ/2TJ can sense the error and inform the user). On this pass, 2TJ also saves the job time limit in PP low core. 2TJ returns control to 1BJ which then completes the setting up of the control point area, i.e. the job time limit is entered (via the monitor function M.NTIME); the exit mode is set (via the monitor function M.REM); and the required CM and ECS field lengths are requested (M.RSTØR). If the necessary field length cannot be granted, 1BJ goes into recall. There are two noteworthy items which follow from this.

(a) All during the course of the NEXT package discussion, 1BJ was making tests to determine if the next task had already been accomplished; such tests were not mentioned for the sake of clarity.

(b) At first glance, it might seem that 1BJ would always obtain storage since it did first check to see that there was enough storage for the job. However, there is no interlock on the value contained in T.UAS and 2 copies of 1BJ might have decided there was adequate storage for their respective jobs; if the sum of the storage requirements exceed available core, then the second 1BJ to request storage will have to wait. Also, a job at another control point might have requested, and been granted, an expanded filed length after 1BJ has read T.UAS and before 1BJ's request for storage.

However, in this case, storage will be granted and 1BJ will drop out (M.DPP).

9. Referring back to DROPPING THE READ PACKAGE, item 3, it can be seen that the identical condition now exists at control point 3, that is, MTR must advance control point 3. (There is a job name; there is no CP or PP activity; there are no PPs in recall or in the delay stack; and there are no pending requests). 1AJ is now loaded into a pool PP and attached to control point 3.

1AJ first checks the error flag which is now not set, so the overlay 2TS (translate statement) is called. As far as 2TS is concerned, there are several major groups of control card types, each of which will be briefly described below. 2TS reads the next control statement from the control card buffer and proceeds in one of the following ways:

(a) SWITCH, MODE, COMMENT, COMMON, RELEASE.

All of these cards require only simple processing and are handled completely within 2TS. When the requested action has been completed and the control card buffer pointers updated, 2TS drops the PP (M.DPP) without returning to 1AJ. This leaves control point 3 ready to be advanced again.

(b) REQUEST

Only a very little initial parameter scanning is done by 2TS. A few parameters are stored in PP low core and 2TS calls REQ (request card processor) by altering the PP input register and returning directly to the PP resident idle loop. (REQ was made into a separate program to avoid duplications since REQ is also callable from a running program) REQ finishes the parameter processing and posts a request for operator action on the console display. It then tests for operator response which naturally will not happen for a period of time measurable in human terms. REQ then simply drops the PP without updating the

control card buffer pointer. (The console message remains, however). This has the effect of leaving the control point ready for advancing but insures that the same REQUEST card will be completely reprocessed. This procedure is reiterated until an operator response is received. REQ then completes processing, updates the control card buffer pointers and then drops the PP; this leaves control point 3 ready to be advanced again.

(c) LØAD, EXECUTE, NØGØ and program call card

Minimal parameter scanning is performed and information is left in PP low core and in locations RA + 2 to RA + 100 within central memory. 2TS then calls LØD into the PP by using the PP resident function R.ØVL. (It might be worth noting that this overlay load unconventionally causes a program - LØD - to be loaded at a lower numbered location than its caller, i.e. LØD is loaded over 1AJ. The tacit assumption is made that LØD is not sufficiently long to overwrite the spot in 2TS from which the call is made; this is a reasonable assumption in this case, but it should not be made generally).

It is not the purpose of this discussion to detail the operation of the relocatable loader; adequate documentation appears elsewhere. A few general remarks will be made, however. The loading process is performed by one or more of the following routines: LØD, LDR, 2LA, 2LB, 2?E, and in the central processor, LØADER, & OVERLØD. The number of routines involved depends upon the type and complexity of the load operation. At the end of loader processing whichever of the 5 PP loader routines is in control drops the PP. If neither a PP or CP execution was initiated during the course of processing (e.g. NØGØ card), then the control point is again ready for advancing.

Otherwise, the control point is considered to be profitably occupied and cannot be advanced until PP and CP activity has ceased.

10. Assuming that the job we have selected for execution at control point 3 contains more than 1 sector of control cards, 1AJ/2TS will eventually be faced with the problem of obtaining the second (or nth) sector. And it is a problem; the method used defies detailed description. In brief, 2TS will detect that it has either a partial control card or none at all to process and so it will return control to 1AJ.

(This and the job termination case are the only instances when 1AJ regains control from 2TS.) 1AJ then calls 1BJ into the same PP by the underhanded method of adding one to the second character of the PP input register and returning directly to the PP resident idle loop. (Note well, anyone who is thinking of whimsically changing the name of either 1AJ or 1BJ!). Now referring back to the NEXT PACKAGE, item 7 and accepting on faith that 1BJ knows how to pick its way through the maze, it can be seen that the next sector of control cards will be read into the control point area. After this operation is completed, 1BJ then drops out, leaving it to MTR to advance the control point.

11. Ultimately, 2TS will detect either an EXIT CARD or the end of the control statements and will return to 1AJ for job termination.

MINATION The following section will occasionally refer back to DROPPING THE READ PACKAGE because some portions of the two procedures are identical. In the current case, however, we are dealing with dropping a user job.

1. Control has just been returned to 1AJ from 2TS. The first decision that is made is that this is the end of the job and not just a request for another control card buffer load. Then the control point priority is tested and it is determined that this is a user job (i.e. it has a

nonzero priority). Next, the control point error flag is checked. Assuming that the job at control point 3 has terminated abnormally for some reason, steps 2, 3 and 4 will be performed; otherwise, 1AJ resumes processing at step 5.

2. 1AJ calls 2CA (checkpoint abort). 2CA checks to see whether or not the job took checkpoint dumps during the course of its execution. If so, all files associated with the job are locked, i.e. left in a state suitable for restarting the job at some later time; if not 2CA simply returns to 1AJ.

3. 2EF is called. In addition to the writing of the error_message (see DROPPING THE READ PACKAGE, item 4) 2EF will locate the file named OUTPUT and will insure that there is an end-of-record mark. Also, 2EF will supply a partial core dump of the control point field length. The latter is accomplished through use of the M.RPJ function (without a delay specified) to load DMP (central memory dump program) into another PP. 2EF waits for the dump to be completed by waiting until the input register of the other PP no longer contains the request for DMP. All this additional processing is done because the control point priority is nonzero.

4. 1AJ then uses 2TS to cycle through the remaining control cards to the next EXIT card or to the end of the statements. If an EXIT card is encountered, processing reverts to normal, i.e. the following cards are processed as though no error had occurred. In the latter case, 1AJ ultimately regains control of the JOB TERMINATION point; this time, however, the error flag will not be set and steps 2, 3 and 4 are skipped.

5. At this point, 1AJ records the running time used at control point 3; all further PP time incurred in terminating the job is considered to be system overhead.

6. 1AJ calls 2CJ (complete job day file) which appends to the file OUTPUT all dayfile messages relevant to the job at control point 3. During the course of execution, these messages were being collected in two places: chronologically within the system dayfile for display purposes and as a permanent record of system activity; and in the job dayfile. Messages are collected in this manner to eliminate the problem of extracting messages belonging to one job when the job terminates.

7. 2CF (close files) is called to search the FNT for files associated with this control point. If a file of name OUTPUT, PUNCH, or PUNCHB is located, the file is assigned a disposition code of print, punch Hollerith or punch binary, respectively. Then, all local files associated with control point 3 which have a nonzero disposition code (including those just set to nonzero) are entered into the output queue. A file is considered to be in the output queue when the following changes have been made to the FNT entry:

   (a) the file name is changed to the job name (as recorded in the control point area)

   (b) the control point assignment is cleared, i.e. set to zero

   (c) the priority level (as recorded in the control point area) with a sublevel of 1 has been entered.

   With one exception, files in the output queue are of type local (local to control point zero). The exception is the file which was named OUTPUT during execution; it is set to type output. The reasons for this are obscure.

8. 1AJ now goes through the process described in item 5 of DROPPING THE READ PACKAGE, i.e. in conjunction with 2DF, it releases files and equipment associated with control point 3. Again it should be noted that the files entered in the output queue were dissociated from the control point and hence, are not affected. Storage is released and t e control point area is cleared.

9. In the case of dropping a system (zero priority) job, 1AJ then dropped out leaving the control point vacant. Now, since it would be a great nuisance to require the operator to bring up NEXT for each job to be run, 1AJ will automatically bring NEXT back to control point 3. This is done on the theory that once a NEXT package resides at a control point, it ought to remain until the input queue is empty. 1BJ is summoned into the same PP by means of the "adding one" method described in item 10 of NEXT PACKAGE; 1BJ begins processing at item 2.

PUT
KAGE

We will now leave control point 3 to the process of selecting and executing jobs and consider the problem of what to do with the output queue.

1. The procedure for bringing up the OUTPUT package is nearly identical with that for bringing up READ and NEXT, so again we shall abbreviate. The operator initiates the process by typing, say 2oOUTPUT. The name of the PP overlay comprising the OUTPUT package is 1ØT. 1ØT will be assigned to a pool PP at control point 2.

2. 1ØT performs the usual initiallization of PP low core (see READ PACKAGE 4), and then enters the job name OUTPUT into the control

point area for display purposes.

3. 1ØT is capable of simultaneously supervising the printing or punching of as many as six files from the output queue, but since each requires the permanent allocation of a 1000_8 word CM buffer (in addition to the base 100_8), 1ØT counts the number of printers and punches to minimise buffer space. The EST is searched (directly by 1ØT) to count the number of equipments of type LP or CP. (Note that these equipments are not reserved at this point). An M.RSTOR function is then issued to MTR for N*1000B + 100B words (maximum 6100B), where N is the number of output devices. This storage will remain assigned for as long as OUTPUT occupies the control point. The base 100_8 words of storage are used to control the ownership of each of the N buffers.

4. 1ØT then checks for a free buffer; since this is the first time through, there will naturally be at least one. The FNT channel is reserved via a monitor function and the FNT is searched for files in the output queue. During the single pass through the FNT, several things occur.

   (a) The print file, if any, with the highest priority is selected for printing. A buffer is assigned to the file and an equipment of type LP is requested from MTR. If either a buffer or a printer cannot be obtained (as can happen at later stages) the file is simply returned to the output queue.

(b) The punch file, if any, with the highest priority is selected for punching. A buffer and a card punch are assigned as in (a) above. The PP occupied by 1ØT is capable of processing only print files, so another PP must be brought into play for a punch file. 1ØT requests (M.RPP) that MTR assign a pool PP to control point 2 for the overlay called 1PØ (punch output).

(c) At an interval selected by the installation, the priority sublevels of all files remaining in the output queue are incremented by one. This insures that for jobs with the same priority level, the output of the eldest in the output queue will be processed first.

(d) Any files selected for processing during this pass must be removed from the output queue; this is accomplished by assigning the FNT entry to, in this case, control point 2.

5. If there were no files selected from the output queue, 1ØT would set the PP recall register and drop out. However, let us assume a print file has been found. 1ØT then calls 3ØT.

6. 3ØT has a few peculiarities which are worth pointing out. 3ØT is called only by 1ØT and in fact, operates as though it were part of 1ØT. It is merely an accident of history that it is not physically a part of 1ØT. Before 3ØT is entered, 1ØT sets some addresses of 1ØT subroutines into PP low core in order that 3ØT may use them. The most noteworthy of these subroutines is the one described in item 4. This subroutine is entered frequently by 3ØT during its processing to insure that printers

and punches are kept as busy as possible. The punch files are farmed out, as mentioned before, and 3ØT keeps track of the progress in printing the several files for which it may be responsible at any one time.

7\. We shall now largely ignore the fact that the printing of other files may be initiated and concentrate on the processing of a single file. However, because several files may be printed at once, the technique used by 1LJ (i.e. that of alternating 2BP/2ES and 2BP/2RC) is not possible for 1ØT/3ØT. The reading of the disc file must be done elsewhere.

The PP overlay CIO was designed primarily to deal with user I/O; however, it may also be called from another PP program provided that user-type parameters are supplied. 1ØT requests from monitor (M.RPP) the assignment of a PP for CIO, supplying for the input register an address within the base 100_8 word area as well as the name, CIO. (As mentioned previously, space is reserved within this area for information about each print file; the first five words for each file are known as a File Environment Table.(FET). The FET format used by 1ØT is identical to that specified for user I/O). CIO independently begins to fill the proper buffer with data from the disc output file and 3ØT loops waiting for completion of the read operation. (During this loop it may also be printing data from other buffers or checking for new files to print or punch). When the buffer is full (or when an-end-of record or end-of-file is read), 3ØT begins to print the data on the printer selected for this buffer, interspersing this process with operations on other buffers and

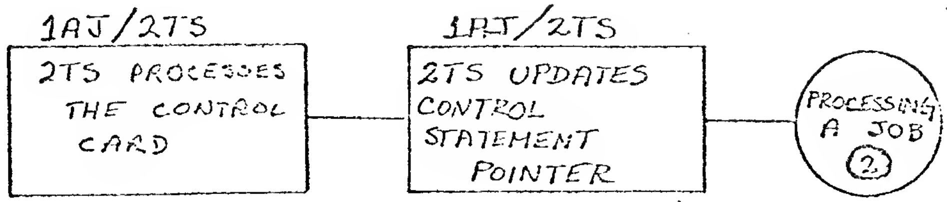
and printers. The printing of the entire file proceeds in this manner. When the end-of-file is reached, 3ØT calls 1CO (complete output) into another PP.

8. 1CO is given the buffer number and the EST ordinal of the printer used. The printer is released (via a monitor function) back to the pool of available equipment. The buffer number is used to locate the relevant FET, and hence the file name; the buffer is then made available for another print or punch file. 1CO then loads 2DF to remove the now defunct disc output file. 1CO then drops out.
(SEE # 6 ON PAGE 19 FOR DAYFILE HANDLING INFORMATION)

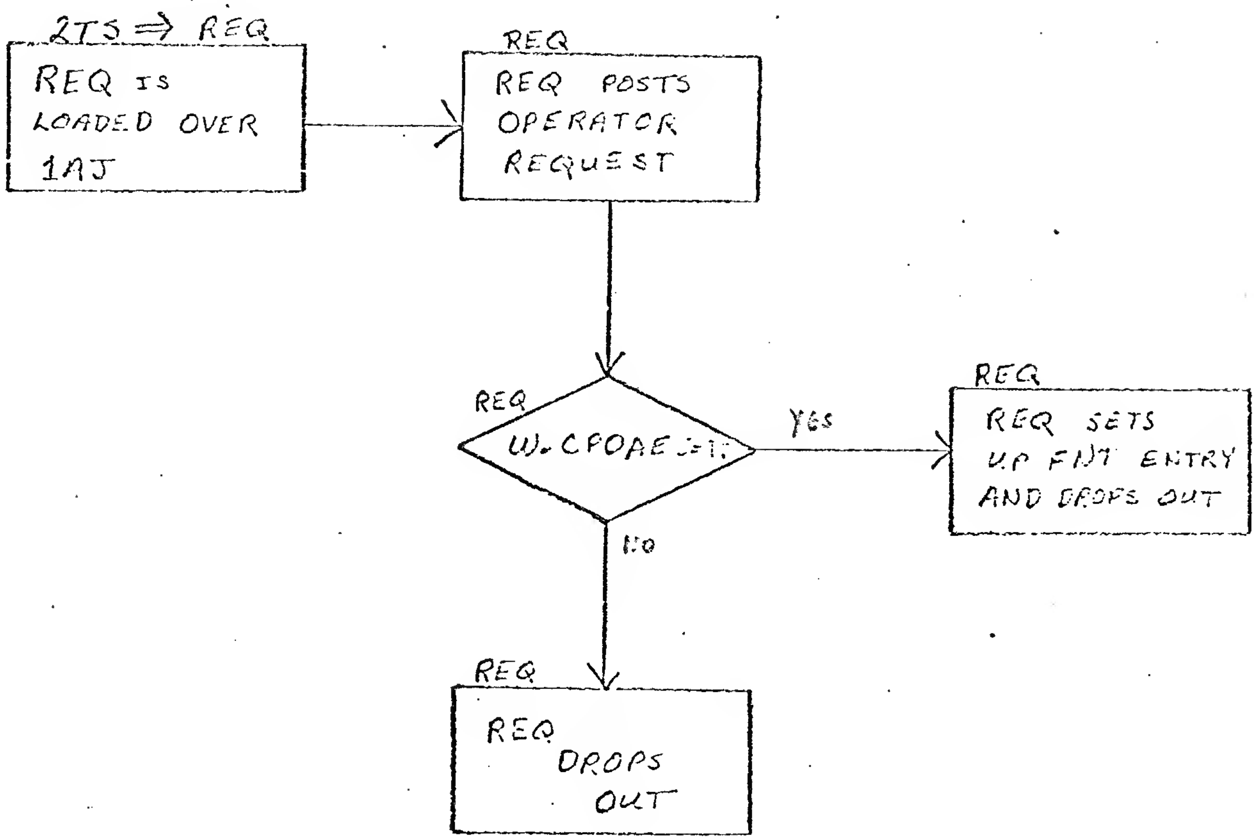
9. Meanwhile, (assuming that a punch file was found) 1PO has been punching cards by alternately calling 2BP/2ES to read the disc file into its CM buffer and 2BP/2PC (punch cards) to punch data from the buffer. CIO is not required since 1PO only handles one punch file at a time; another PP and another copy of 1PO would be required if two punches were to operate simultaneously. When the end-of-file is reached, 1PO itself performs the actions described in (8) above, calling 2DF to remove the disc punch file. 1PO then drops out.

29th September, 1967.

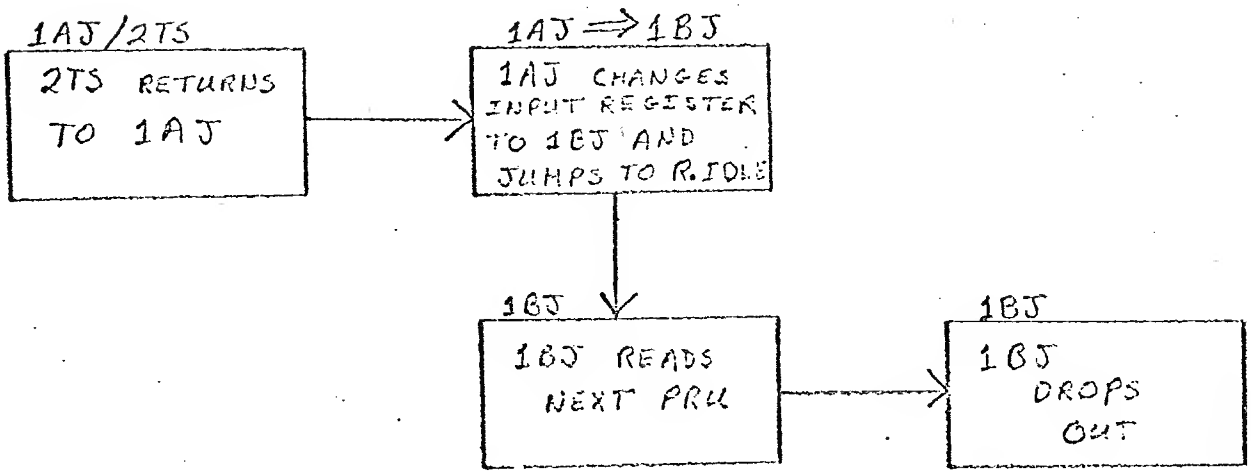
STANDARD CONTROL CARD



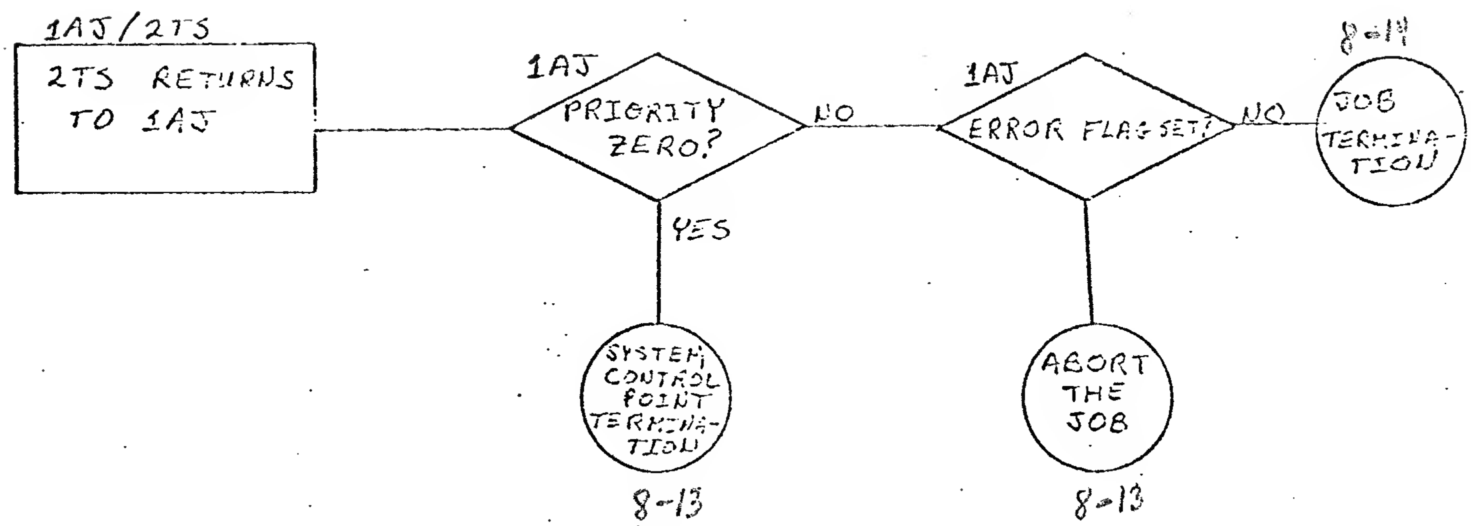
REQUEST CARD



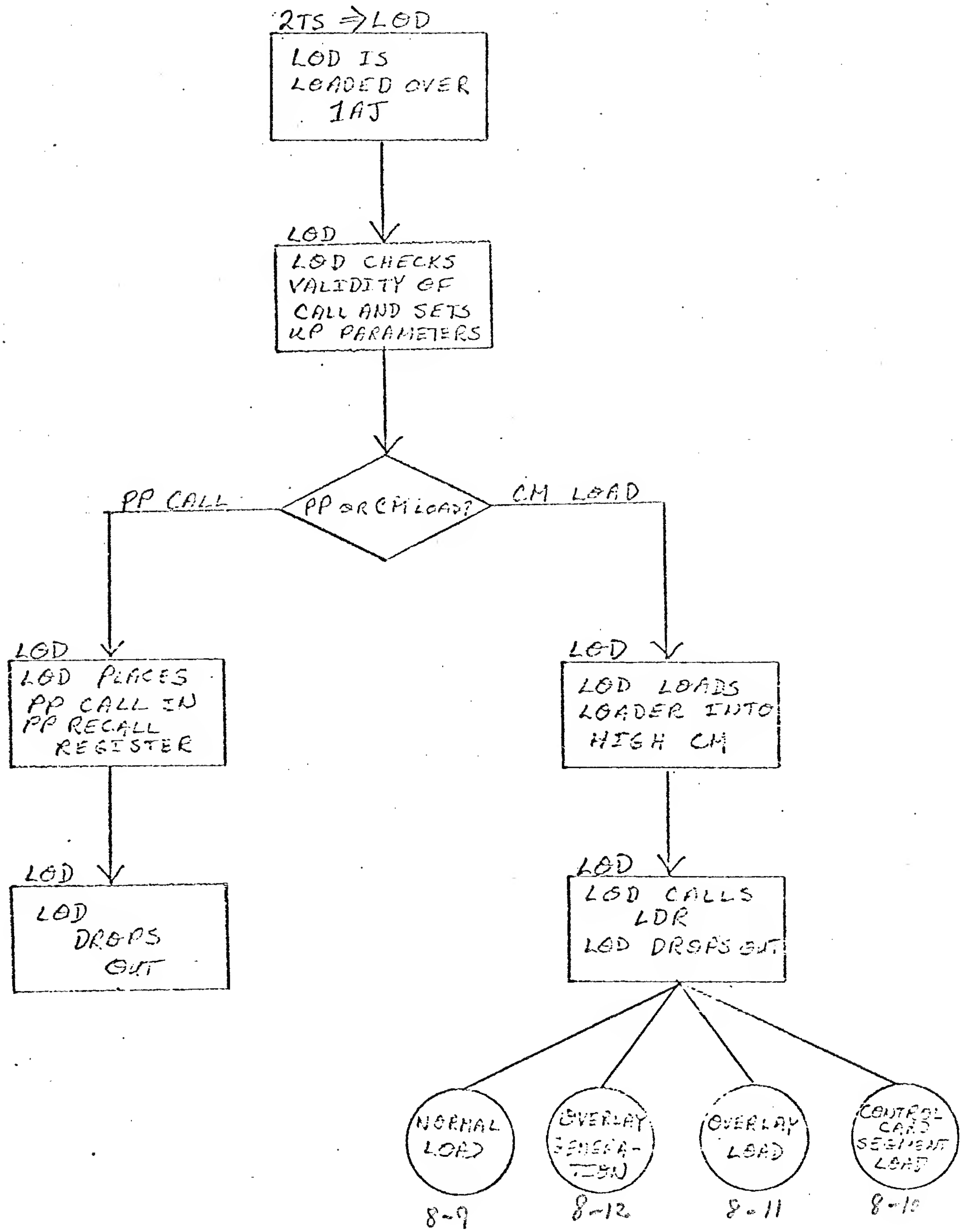
CONTROL CARD BUFFER EXHAUSTED



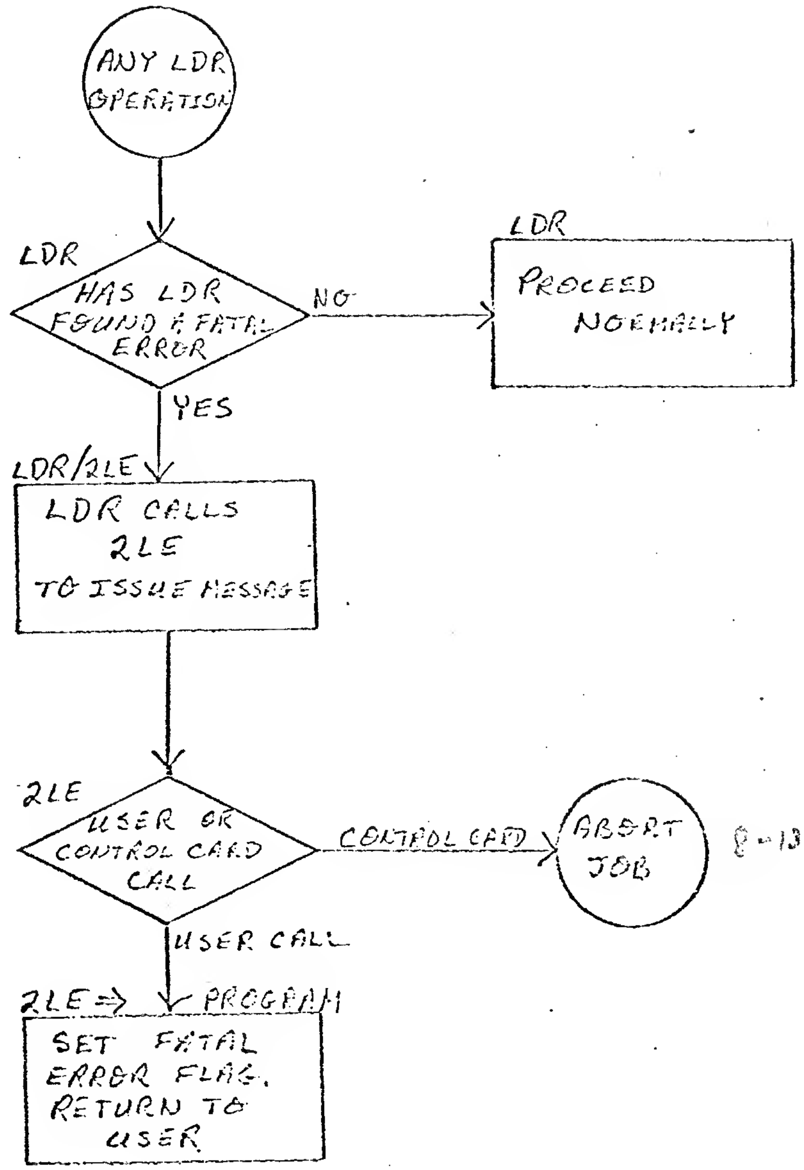
NO MORE CONTROL CARDS



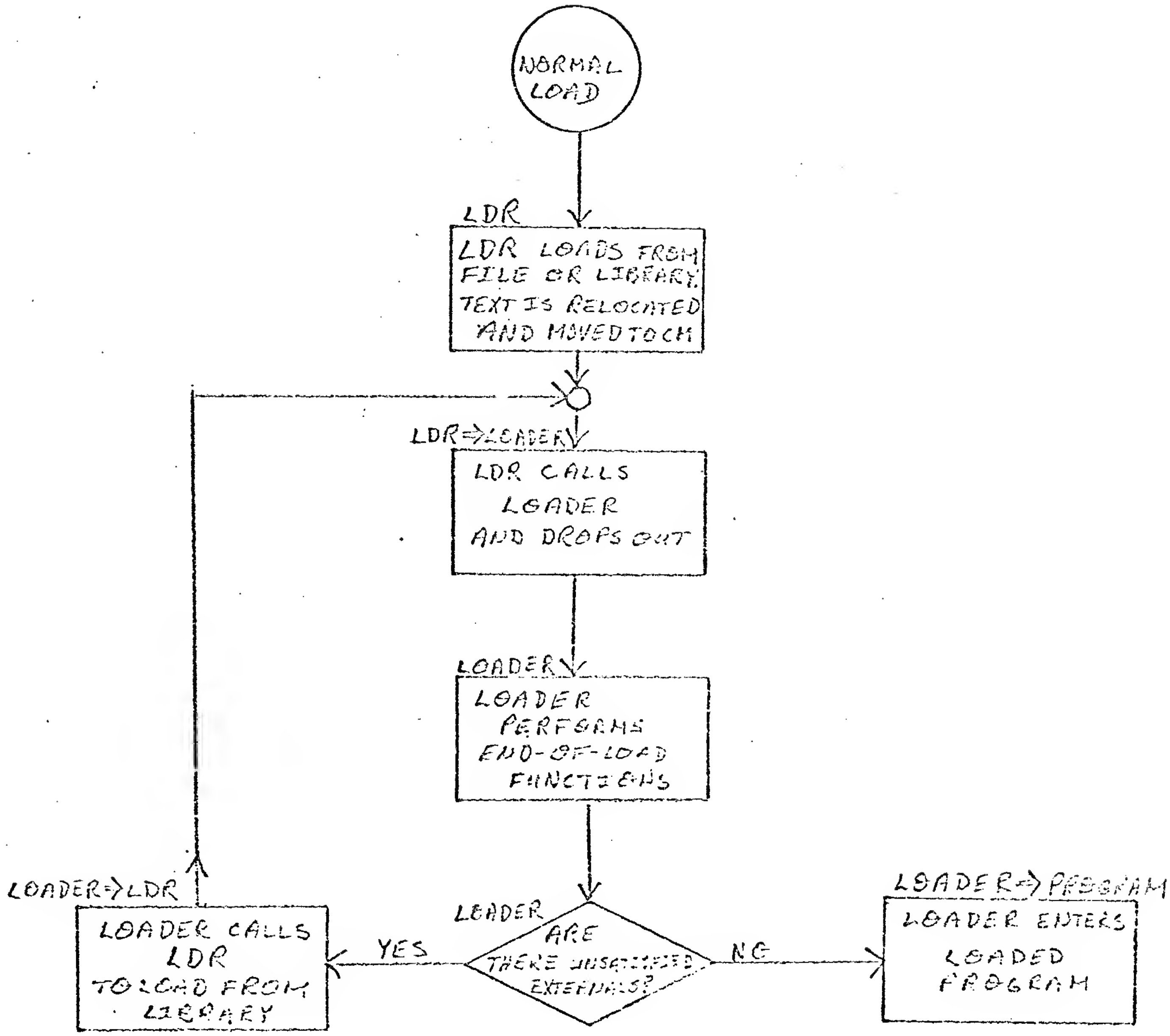
LOADER CALL


2TS ⇒ LOD
LOD IS LOADED OVER 1AJ
LOD
LOD CHECKS VALIDITY OF CALL AND SETS UP PARAMETERS
PP OR CM LOAD?
PP CALL
CM LOAD
LOD
LOD PLACES PP CALL IN PP RECALL REGISTER
LOD
LOD DROPS OUT
LOD
LOD LOADS LOADER INTO HIGH CM
LOD
LOD CALLS LDR
LOD DROPS OUT
NORMAL LOAD
OVERLAY GENERATION
OVERLAY LOAD
CONTROL CARD SEGMENT LOAD
8-9
8-12
8-11
8-10

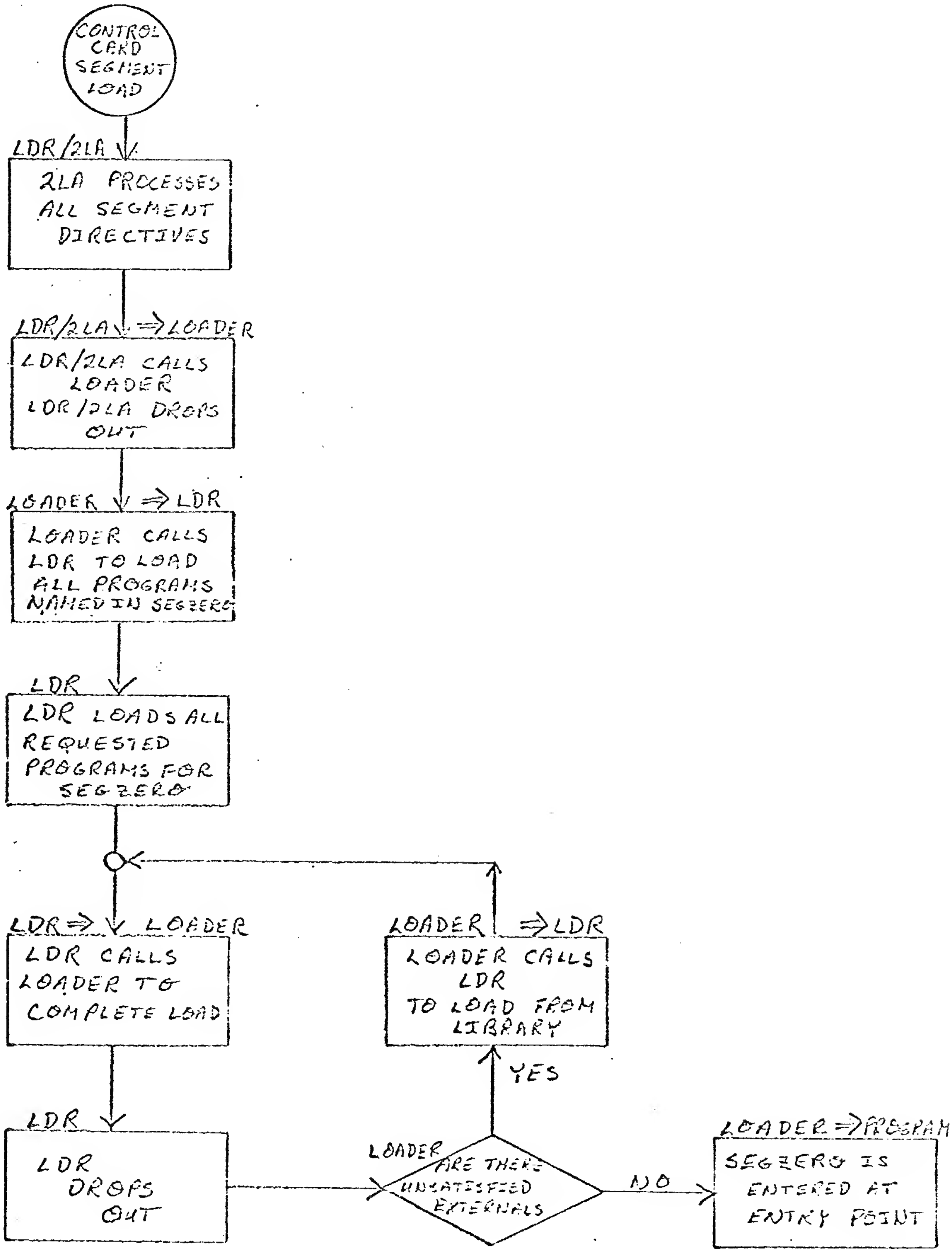
LOADER CALL (CONTINUED)



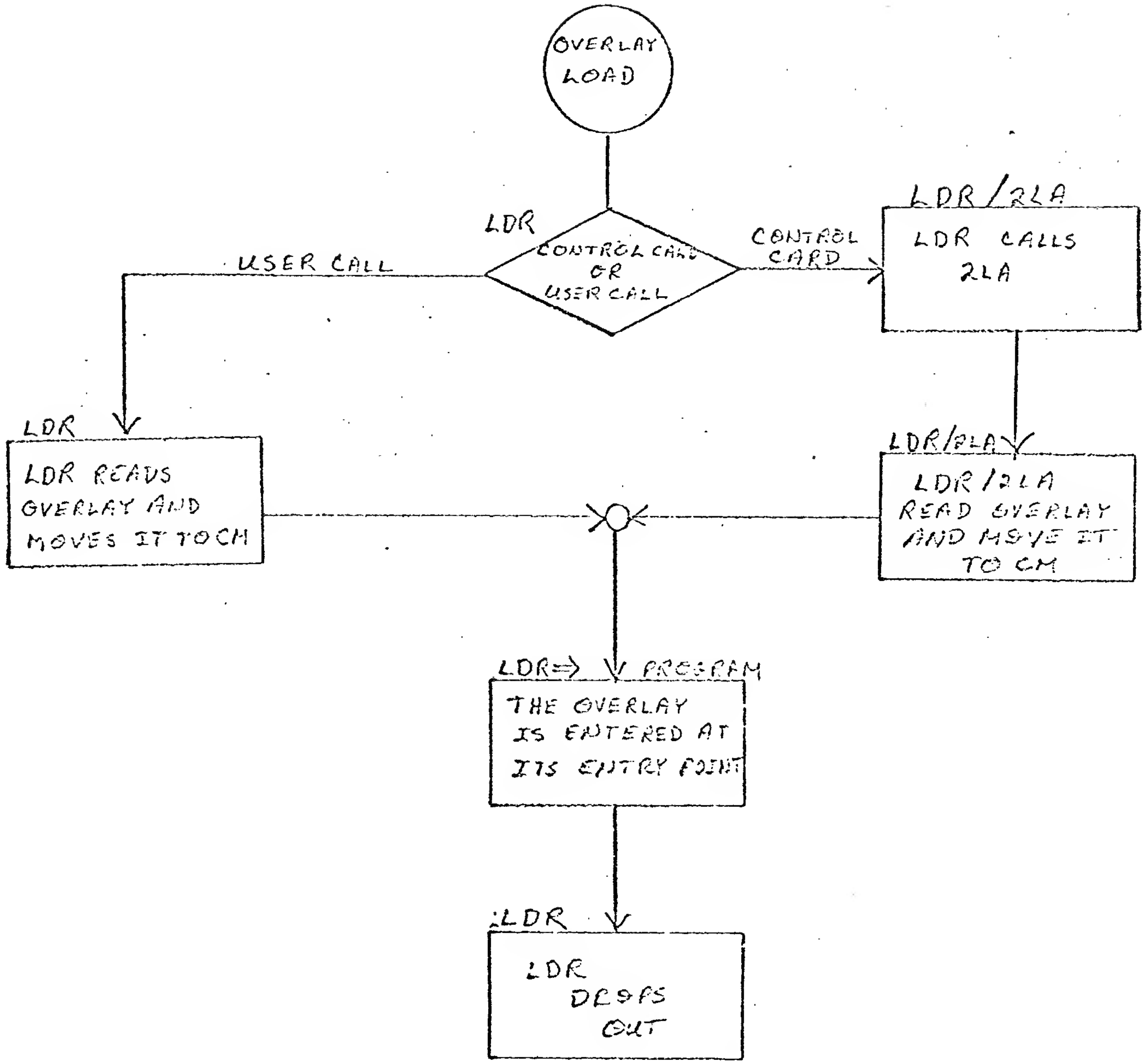
LOADER CALL (CONTINUED)



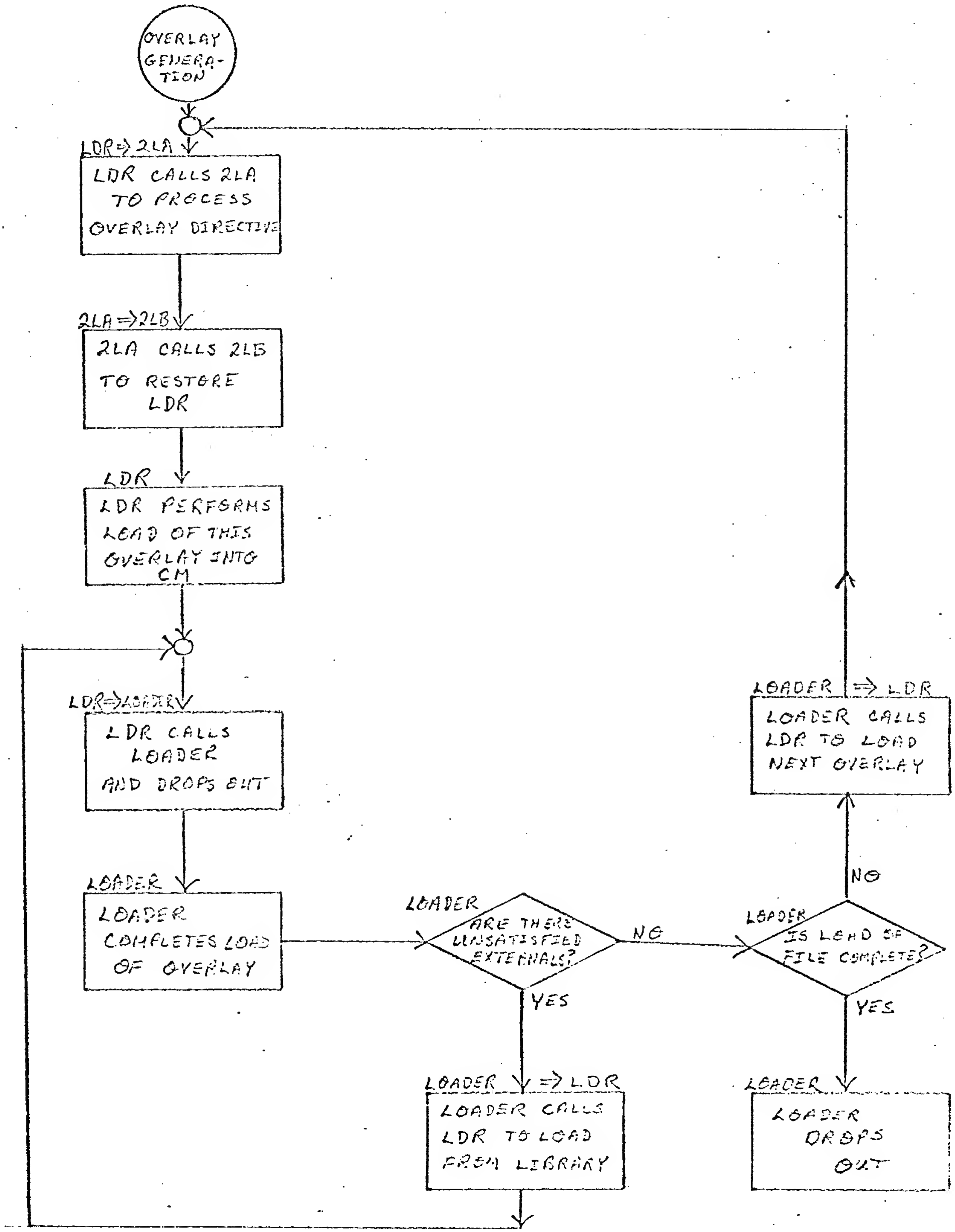
LOADER CALL (CONTINUED)



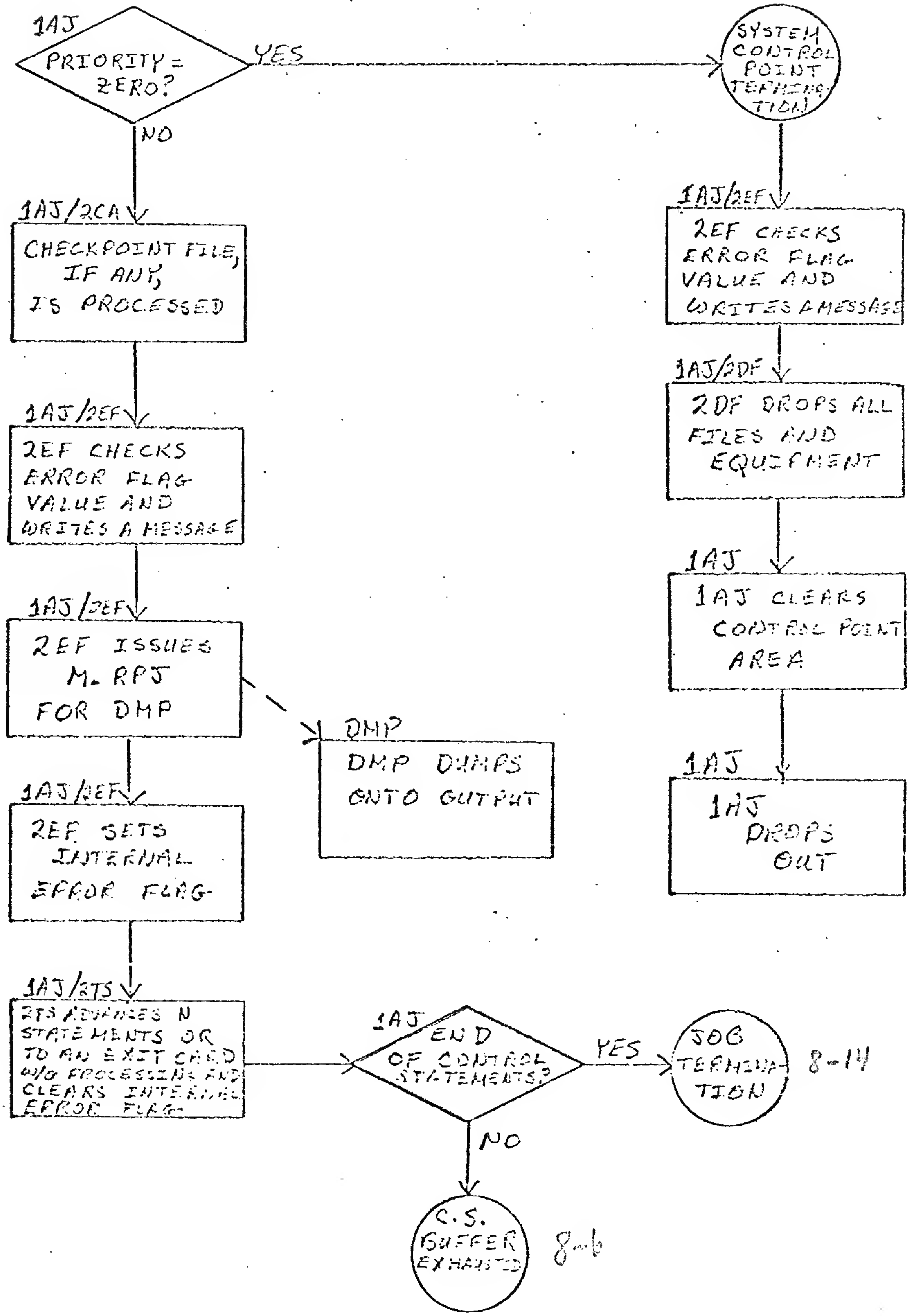
LOADER CALL (CONTINUED)



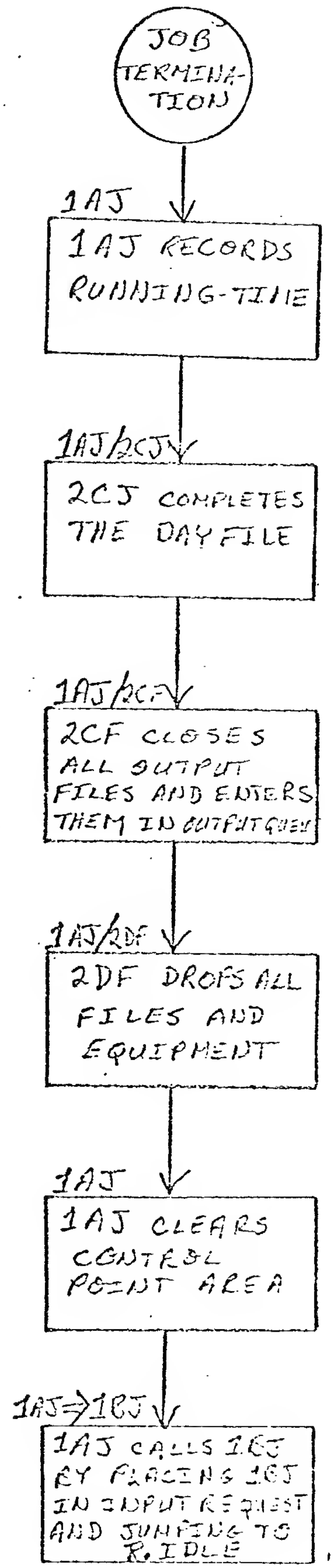
LOADER CALL (CONTINUED)



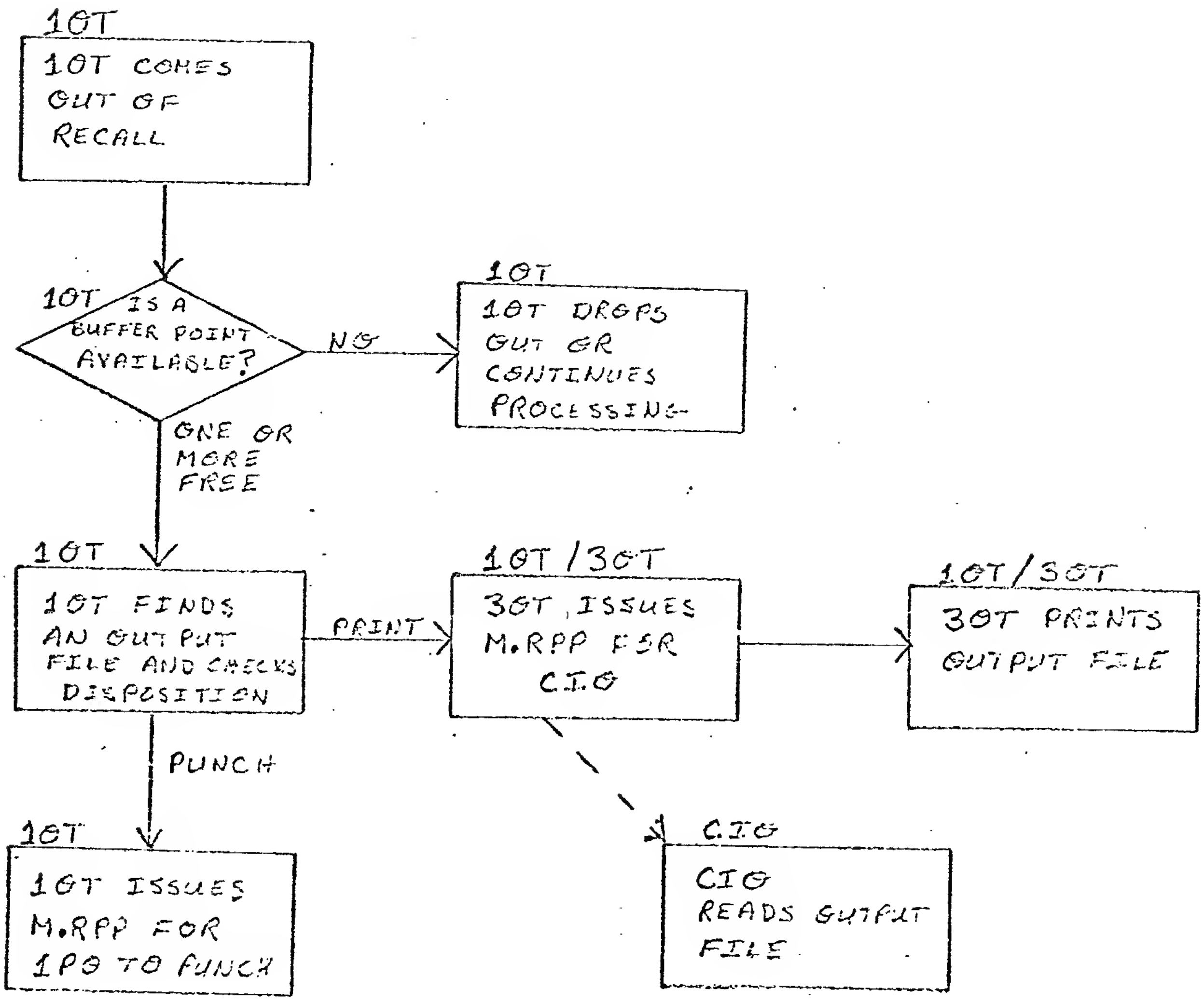
ABORTING A JOB

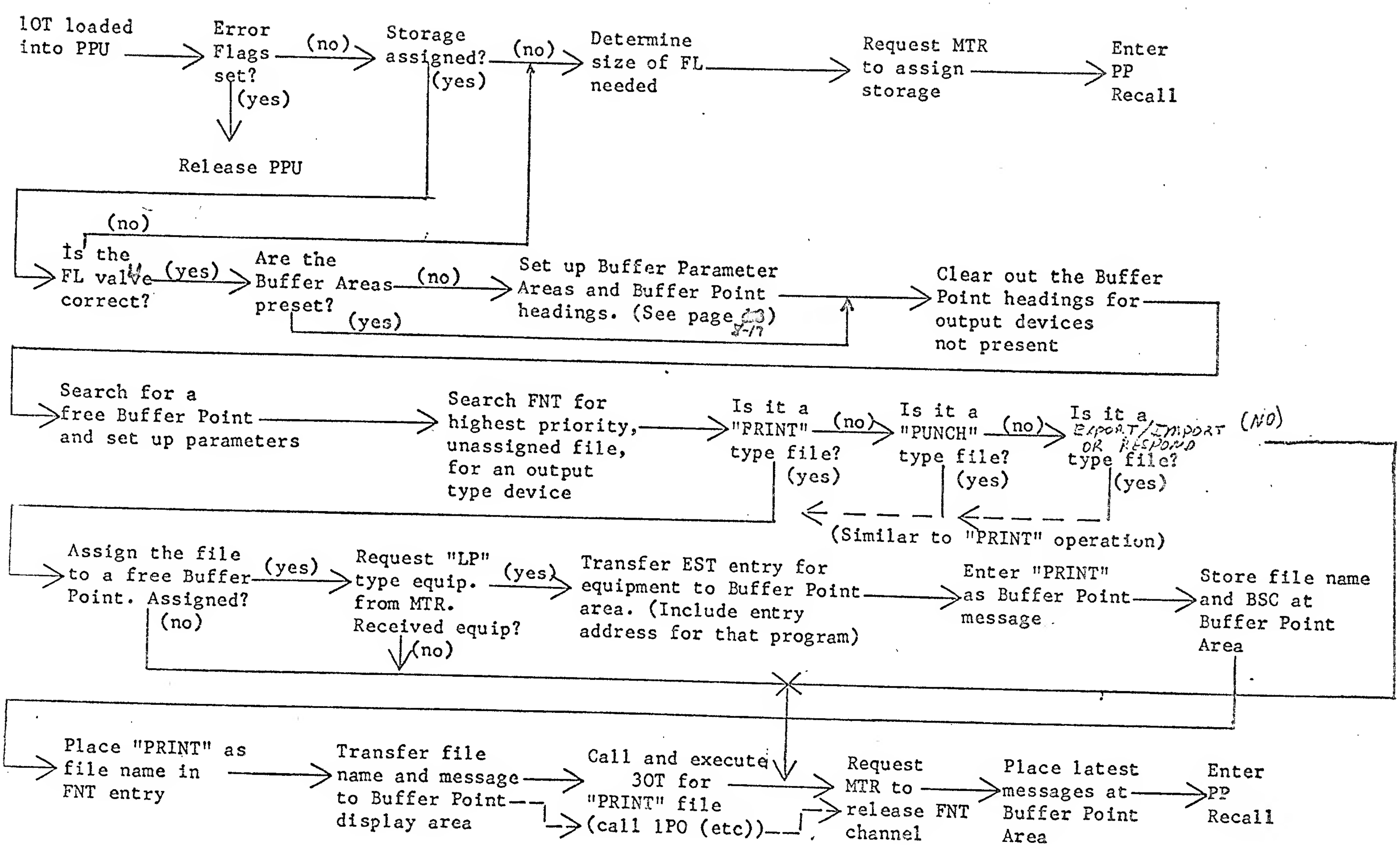


ABORTING A JOB (CONTINUED)



PROCESSING OUTPUT FROM A JOB
(DUMP JOB)





"OUTPUT" SIMPLIFIED FLOW CHART

101

OAD JOB

LJ → 2RC → 1LJ → 2TJ (AT 6400B) → 1LJ → 2BP → 1LJ → 2

CHECK TO SEE
IF SOMETHING IS
IN BUFFER AREA

- CODE + STATUS TO FET
- JOB NAME TO FET
- JOB NAME TO CP

- ZERO OUT
ENTRY AREA
- LOAD 2ES

T UP
ET

- READ 1 CARD
IN HOLLERITH
- CONVERT HOL → EXT
BCD → DPC; THEN
PACKED INTO BYTES
WRITE TO CM
- LOOP UNTIL EOR/EOF/Buffer full

- TRANSLATE JOB CARD
- BUILDING FNT ENTRY
- BUILD PRIORITY SCHEDULE

- VERIFY FILE NAME
- VERIFY BUFFER PARAMETERS
* - ENTER FILE NAME AND CODE
+ STATUS INTO A FREE FNT ENTRY (000024
* - ENTER FILE AS A "LOCAL",
ASSIGNING TO CP1
* - CLEAR FNT (2)
* - SET FNT (3) = 0 ——— 01
* - ENTER FET ADDRESS IN FNT(3)
- SET UP DEVICE TYPE IN FET

* = 1st PASS ONLY

A = 0 ——— 0
A+7 = 0 ——— 0
A+10 = ABC ——— REQ WRITE
11 = 000 004 01 ——— 0100 = FIRST
12 = 0 ——— 205 = IN
13 = 0 ——— 00100 = OUT
14 = 0 ——— 2100 = LIMIT
15 =
UNCHANGED
20 =
21 3100
22 3100
23 3100
24 3300
3100

RA+100 = 1ST CARD (10 CHAR)
101 = ABC, P7, T100, CM400000.
102 = 0 ——— 0.

→ 2ES (AT 50008) → 1LJ → 2BP → 1LJ

2ES (AT 50008)

- * – ASSIGN AN RB TO FILE FROM BEST RBR
- * – ASSIGN AN RBT CHAIN TO FILE
- – BUILD STACK ENTRY
- – REQUEST STACK ENTRY OF MTR (MTR ENTERS THE REQUEST AND CALLS A STACK PROCESSOR)

* = 1ST PASS ONLY

1LJ

- – PAUSE FOR STORAGE RELOCATION
- – WAIT UNTIL FILE IS WRITTEN ONTO DISK
- – RESET FET NAME TO "READ"

2BP

- – VERIFY FILE NAME
- – VERIFY BUFFER PARAMETERS
- – RESERVE FILE

1LJ

- – SET DEVICE TYPE CODE TO C.R. (60)

→ 2RC → 1LJ → EOF? — YES → GO TO RECALL; NO → 101

2RC

- – READ 1 CARD.
- – CONVERT FROM HOLLERITH → EXT. BCD → DPC.
- – BUILD CM WORD.
- – WRITE TO CM BUFFER AREA.
- – LOOP TIL EOR/EOF/BUFFER FULL

1LJ

- – UPDATE STATUS IN FET + FNT

EOF — YES

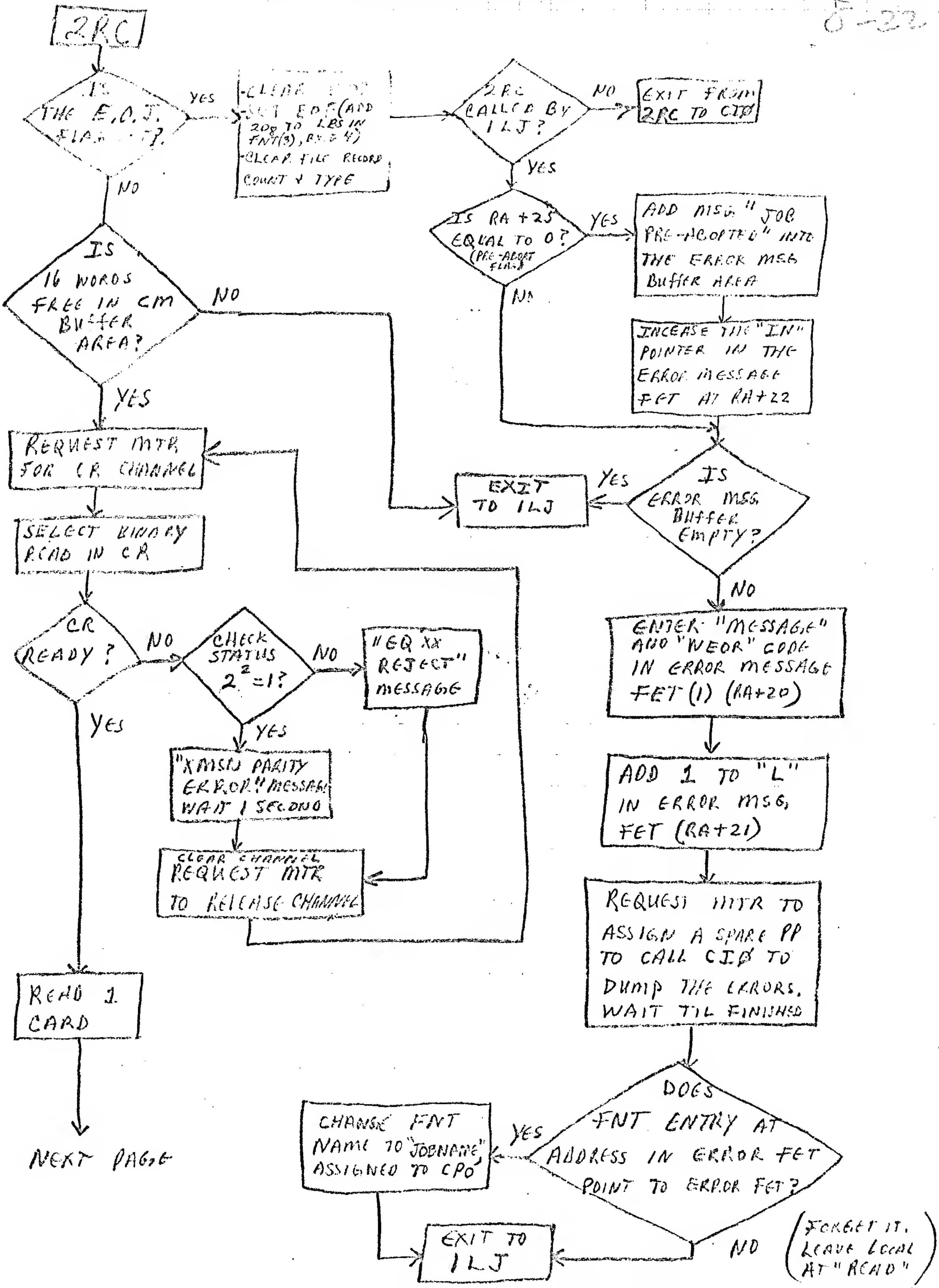
- – UPDATE PP TIME
- – ENTER DAYFILE ENTRY " READ. "PP TIME"
- – PLACE FILE NAME IN FET
- – EOR LEVEL TO FET

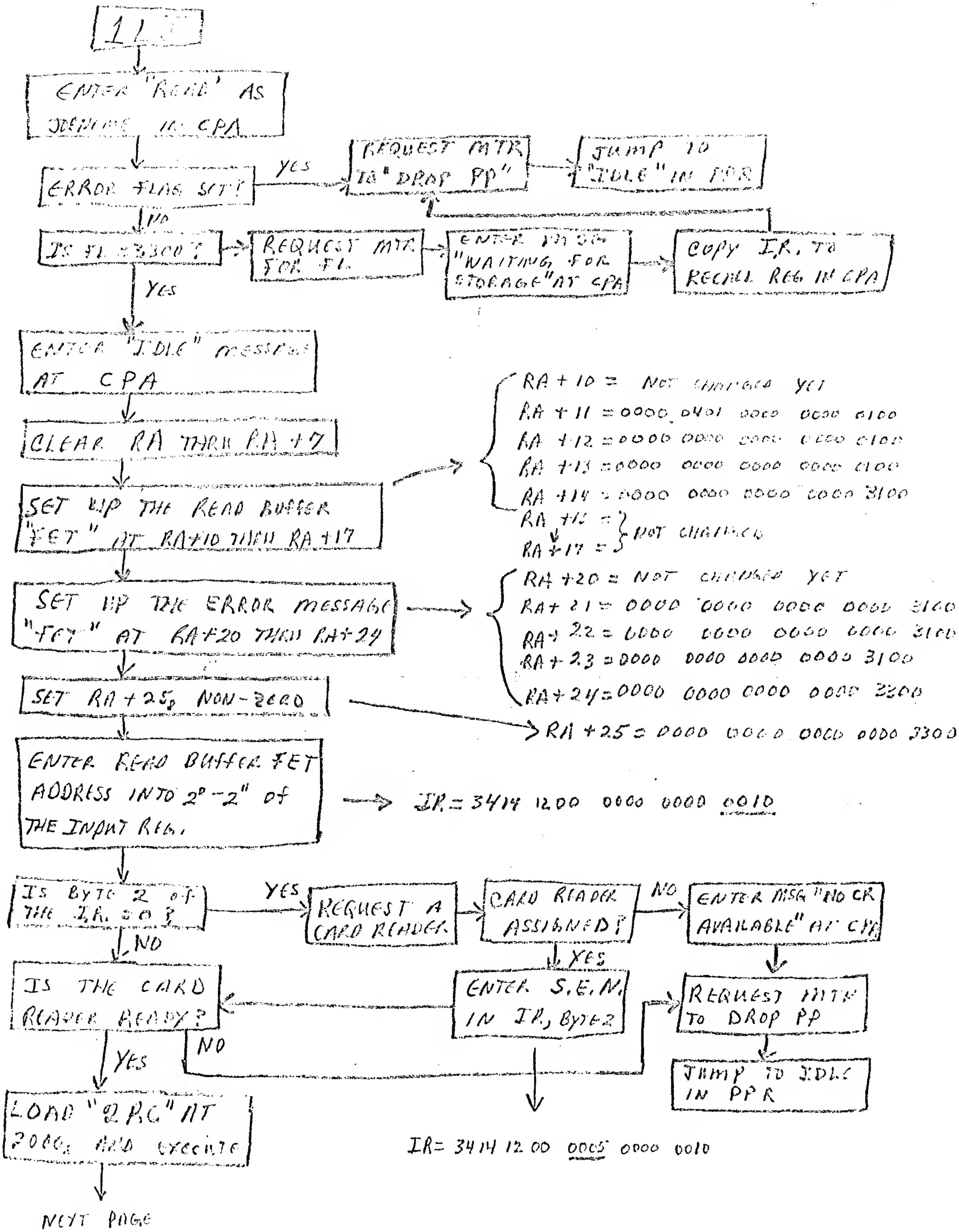
EOF — NO

101

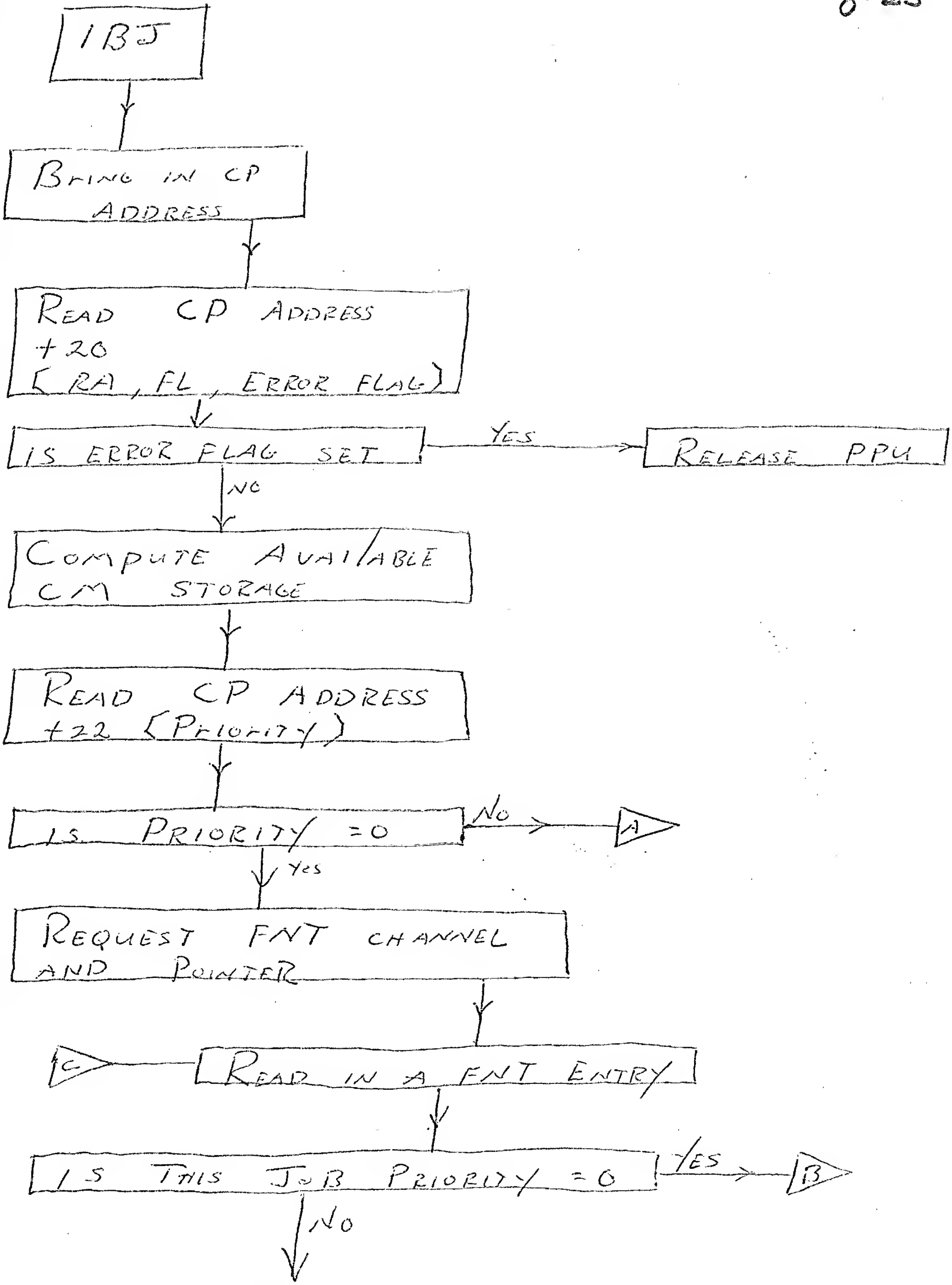
2ES Parameters

	PP Call	
D.FNT	20	
	21	
	22	FNT(2)
	23	
	24	
	25	
	26	FNT(3)
	27	
	30	← level number in low 6 bits — Also to FNT(3)
	31	← OP Code
D.EST	32	Flags 4=NOFET, 1=PP AVAIL
	33	CP RECALL 1=RECALL, 0=NO RECALL (Ignored if D.CPAD=0)
D.BA	40	First word of FET, 2nd word of FET
	41	If NO FET BIT If open function
	42	Is not set (for random bit)
	43	
	44	
D.FR5	45	Last buffer status
D.PPIRB	50	
	51	
	52	
	53	Number of records
	54	FET address
D.FA	57	
D.FIRST	60	First
	61	
D.LIMIT	66	Limit
	67	
D.CPAD	74	Control point address


2RC
IS THE E.O.J. FLAG SET?
YES
NO
-CLEAR EOJ -SET EOF (ADD 20₈ TO LBS IN FNT(3), BYTE 4) -CLEAR FILE RECORD COUNT & TYPE
2RC CALLED BY ILJ?
NO
EXIT FROM 2RC TO CIO
YES
IS RA+25 EQUAL TO 0? (PRE-ABORT FLAG)
YES
ADD MSG "JOB PRE-ABORTED" INTO THE ERROR MSG BUFFER AREA
NO
INCREASE THE "IN" POINTER IN THE ERROR MESSAGE FET AT RA+22
IS 16 WORDS FREE IN CM BUFFER AREA?
NO
YES
REQUEST MTR FOR CR CHANNEL
EXIT TO ILJ
YES
IS ERROR MSG BUFFER EMPTY?
NO
SELECT BINARY READ IN CR
CR READY?
NO
CHECK STATUS 2² = 1?
NO
"EQ XX REJECT" MESSAGE
YES
YES
"XMSN PARITY ERROR" MESSAGE WAIT 1 SECOND
CLEAR CHANNEL REQUEST MTR TO RELEASE CHANNEL
ENTER "MESSAGE" AND "WEOR" CODE IN ERROR MESSAGE FET (1) (RA+20)
ADD 1 TO "L" IN ERROR MSG FET (RA+21)
REQUEST MTR TO ASSIGN A SPARE PP TO CALL CIO TO DUMP THE ERRORS. WAIT TIL FINISHED
READ 1 CARD
NEXT PAGE
DOES FNT ENTRY AT ADDRESS IN ERROR FET POINT TO ERROR FET?
YES
CHANGE FNT NAME TO "JOBNAME" ASSIGNED TO CPO
NO
(FORGET IT. LEAVE LOCAL AT "READ")
EXIT TO ILJ



11
ENTER "READ" AS JOBNAME IN CPA
ERROR FLAG SET?
YES
REQUEST MTR TO "DROP PP"
JUMP TO "IDLE" IN PPR
NO
IS FL ≥ 3300?
REQUEST MTR FOR FL
ENTER MSG "WAITING FOR STORAGE" AT CPA
COPY I.R. TO RECALL REG IN CPA
YES
ENTER "IDLE" MESSAGE AT CPA
CLEAR RA THRU RA+7
SET UP THE READ BUFFER "FET" AT RA+10 THRU RA+17
RA + 10 = NOT CHANGED YET
RA + 11 = 0000 0401 0000 0000 0100
RA + 12 = 0000 0000 0000 0000 0100
RA + 13 = 0000 0000 0000 0000 0100
RA + 14 = 0000 0000 0000 0000 3100
RA + 15 =
RA + 17 = NOT CHANGED
SET UP THE ERROR MESSAGE "FET" AT RA+20 THRU RA+24
RA + 20 = NOT CHANGED YET
RA + 21 = 0000 0000 0000 0000 3100
RA + 22 = 0000 0000 0000 0000 3100
RA + 23 = 0000 0000 0000 0000 3100
RA + 24 = 0000 0000 0000 0000 3300
SET RA + 25₈ NON-ZERO
RA + 25 = 0000 0000 0000 0000 3300
ENTER READ BUFFER FET ADDRESS INTO 2⁰-2¹ OF THE INPUT REG.
IR = 3414 1200 0000 0000 0010
IS BYTE 2 OF THE I.R. = 0?
YES
REQUEST A CARD READER
CARD READER ASSIGNED?
NO
ENTER MSG "NO CR AVAILABLE" AT CPA
YES
ENTER S.E.N. IN IR, BYTE 2
REQUEST MTR TO DROP PP
JUMP TO IDLE IN PPR
NO
IS THE CARD READER READY?
NO
YES
LOAD "2RC" AT 2000₈ AND EXECUTE
IR = 3414 12 00 0005 0000 0010
NEXT PAGE



IBJ
BRING IN CP ADDRESS
READ CP ADDRESS +20 [RA, FL, ERROR FLAG]
IS ERROR FLAG SET
YES
RELEASE PPU
NO
COMPUTE AVAILABLE CM STORAGE
READ CP ADDRESS +22 (PRIORITY)
IS PRIORITY =0
NO
A
YES
REQUEST FNT CHANNEL AND POINTER
C
READ IN A FNT ENTRY
IS THIS JOB PRIORITY =0
YES
B
NO

IS IT AN UNASSIGNED INPUT FILE — No → B

Yes

WILL IT FIT IN AVAILABLE MEMORY — Yes → B

No

B → SAVE PRIORITY AND FNT ADDRESS → ADVANCE FNT SEARCH ADDRESS

ARE THERE MORE FNT ENTRIES — Yes → C

No

IS THERE AN UNASSIGNED INPUT FILE

YES → CLEAR FL & REWIND FILE IN FST

NO → RELEASE FNT CHANNEL, STOR "NEXT" DISPLAY "IDLE", RELEASE PP

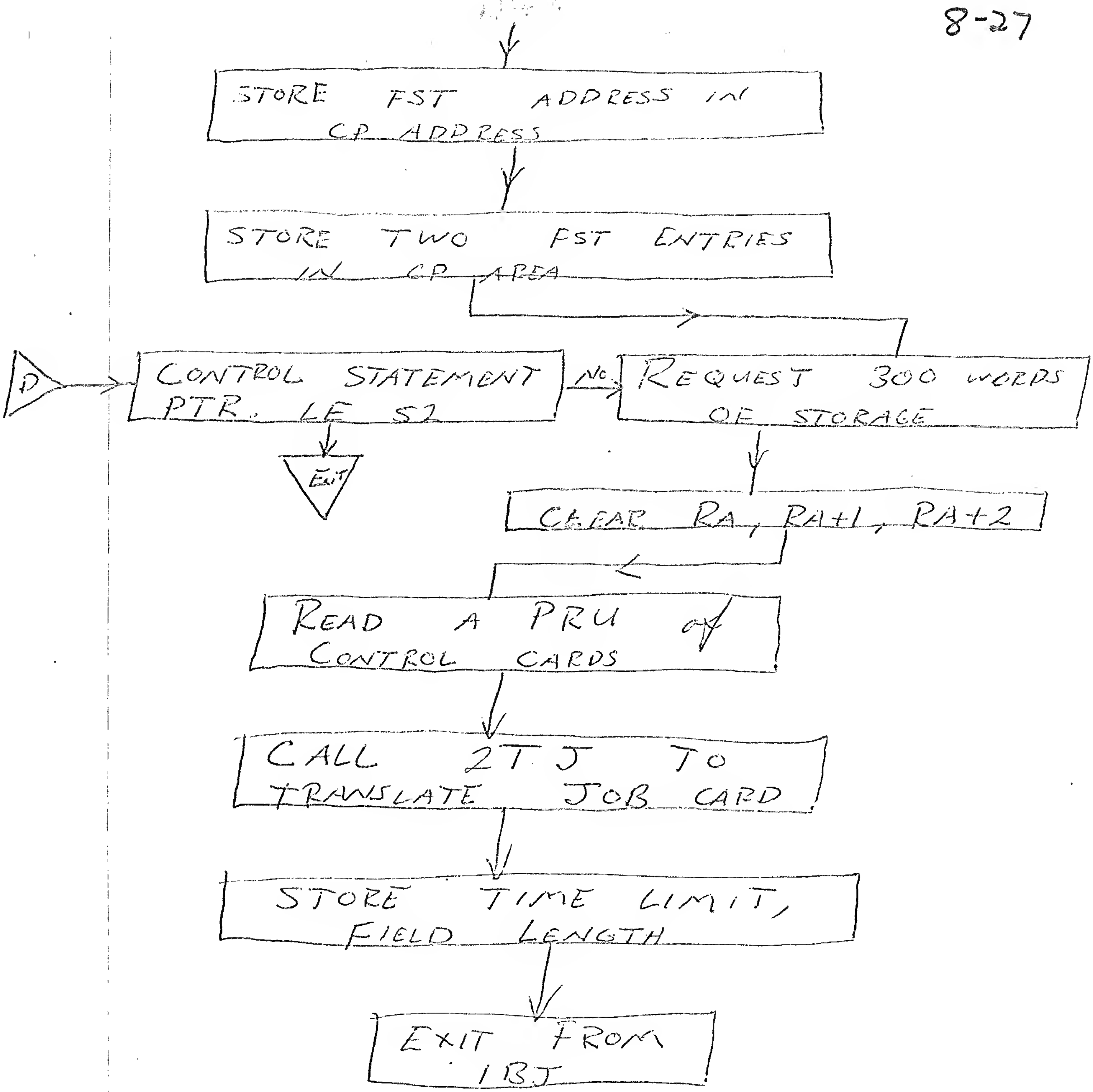
REQUEST PRIORITY

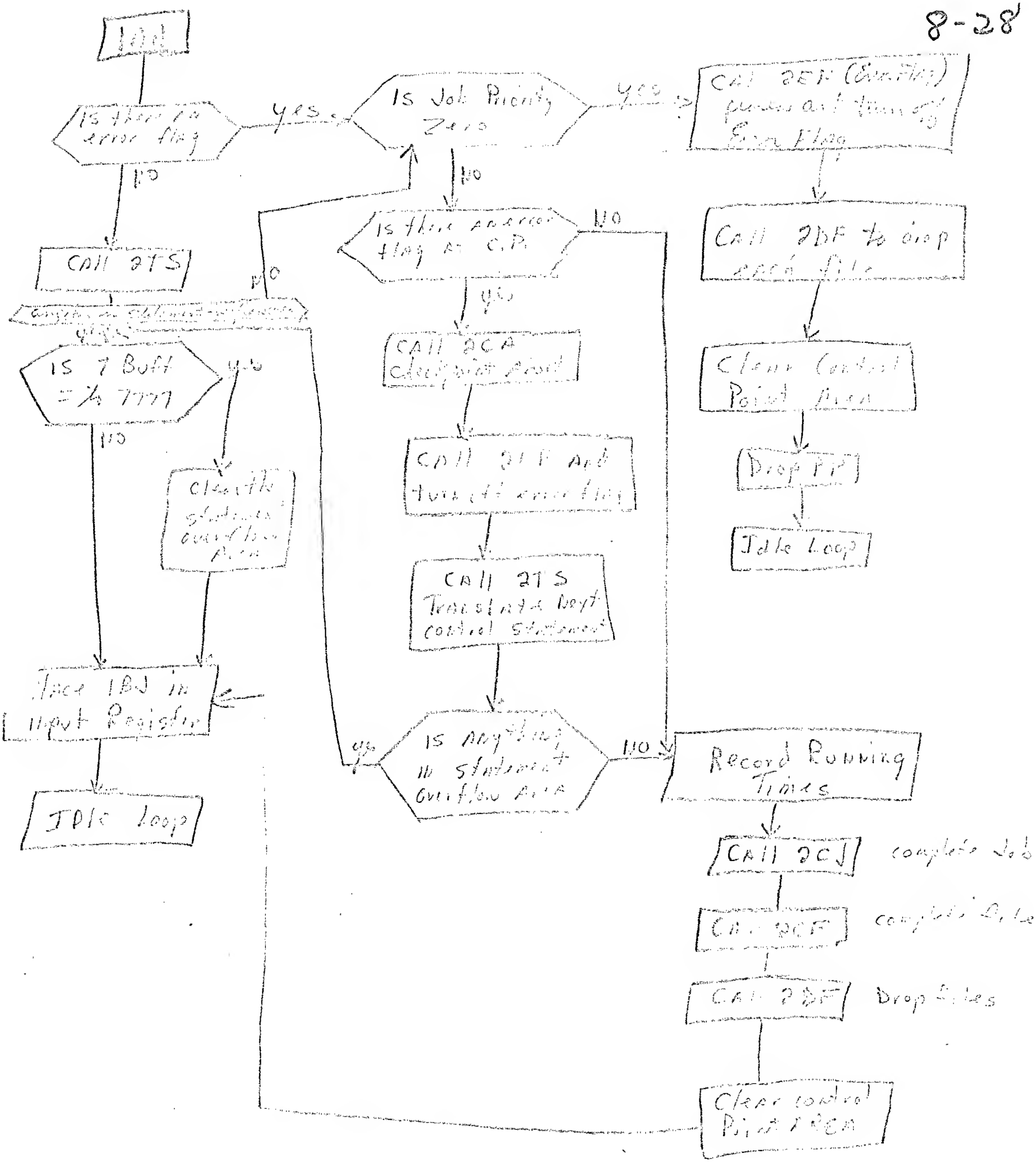
STOR JOB NAME IN CP AREA
SET FILE TYPE TO LOCAL
SET NAME TO INPUT
DROP FNT CHANNEL

A → IS CONTROL STATEMENT POINTER $\neq 0$ (HAVE JOB CARDS BEEN LOADED)

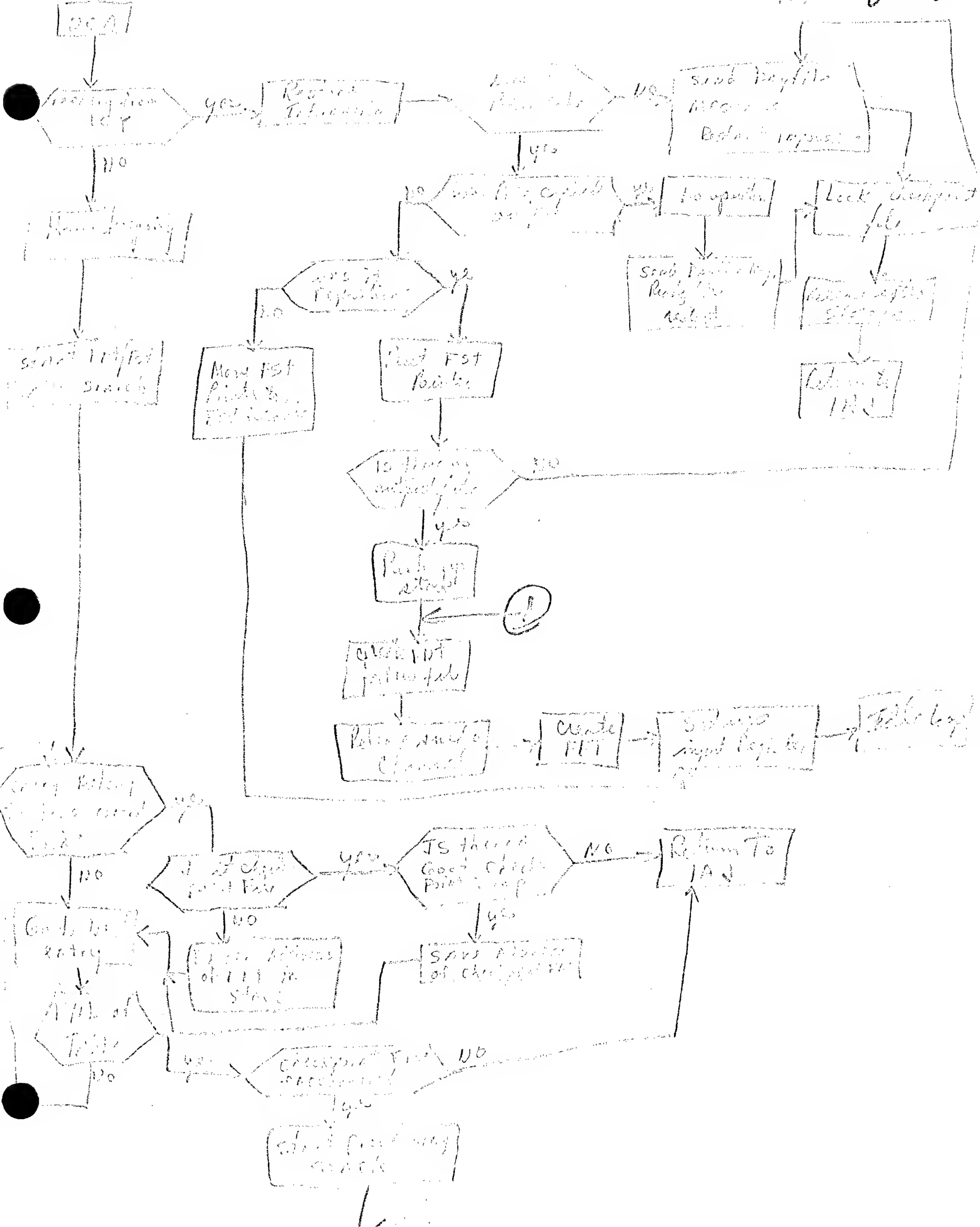
NO

Yes → D

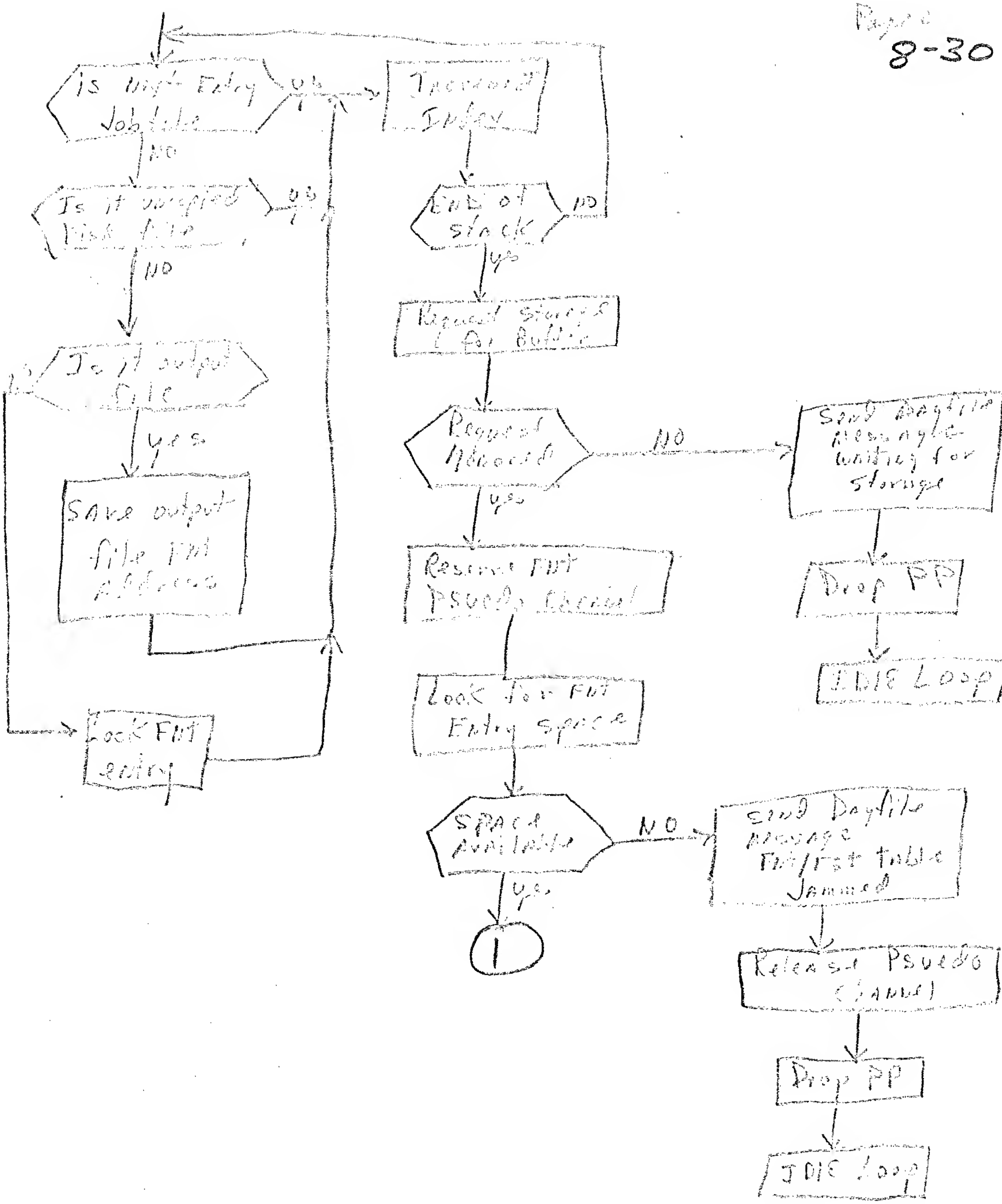

STORE FST ADDRESS IN CP ADDRESS
STORE TWO FST ENTRIES IN CP AREA
D
CONTROL STATEMENT PTR. LE 52
NO
EXIT
REQUEST 300 WORDS OF STORAGE
CLEAR RA, RA+1, RA+2
READ A PRU OF CONTROL CARDS
CALL 2TJ TO TRANSLATE JOB CARD
STORE TIME LIMIT, FIELD LENGTH
EXIT FROM 1BJ

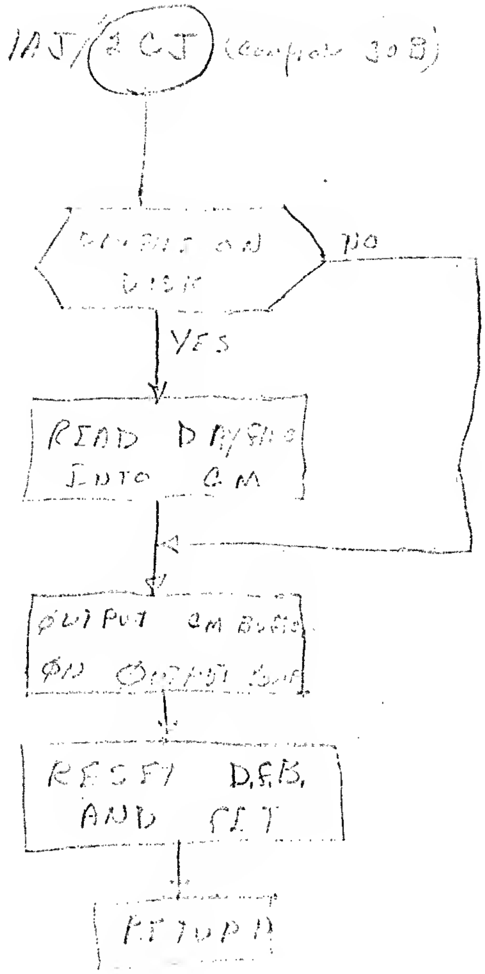

Idel
Is there an error flag
yes
no
Is Job Priority Zero
yes
Error Flag
no
Call ?TS
Call ?DF to drop each file
Is there an error flag at C.P.
no
yes
Is 7 Buff = 7777
yes
no
Clear Control Point Area
Call ?CA
Drop PP
Clear the statement overflow area
Call ?EF and turn off error flag
Idle Loop
Call ?TS
Translate next control statement
Place IBJ in input Register
yes
Is anything in statement overflow area
no
Record Running Times
IDle loop
Call ?CJ
complete Job
complete file
Drop files
Clear control Point area

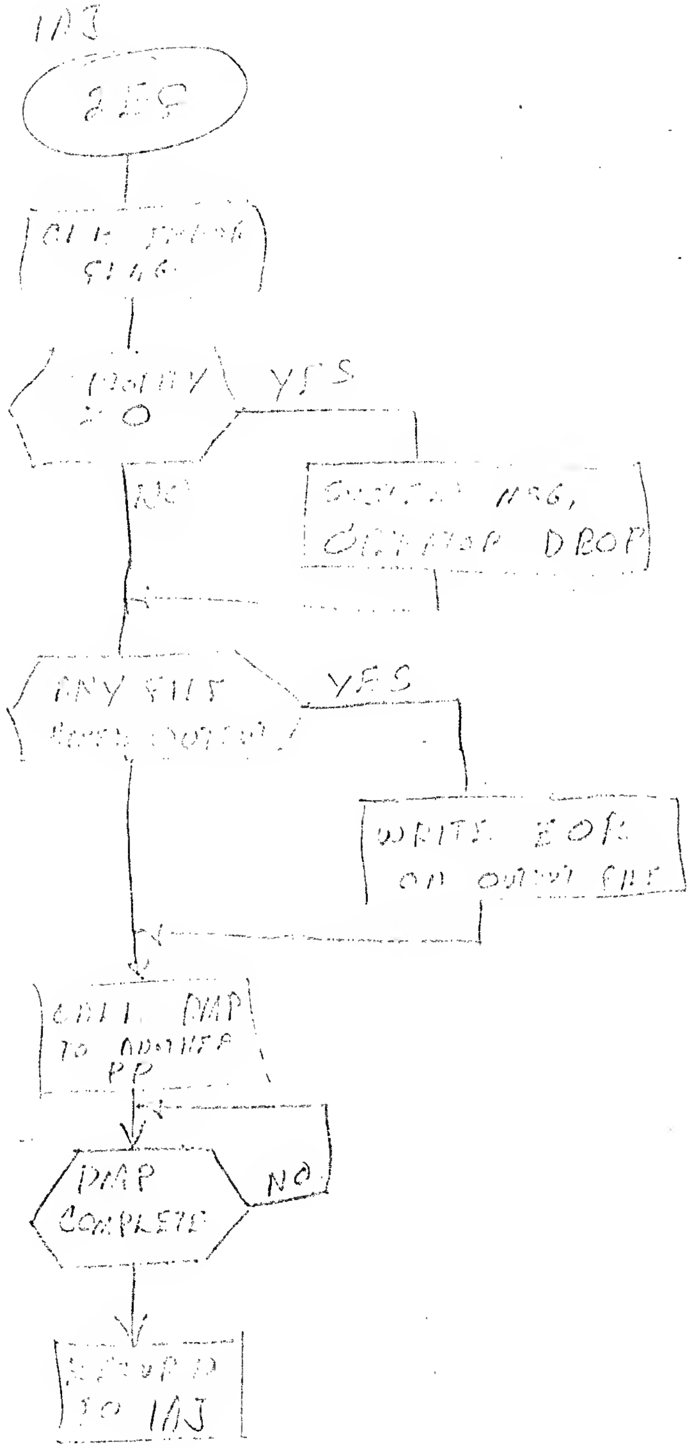
8-29



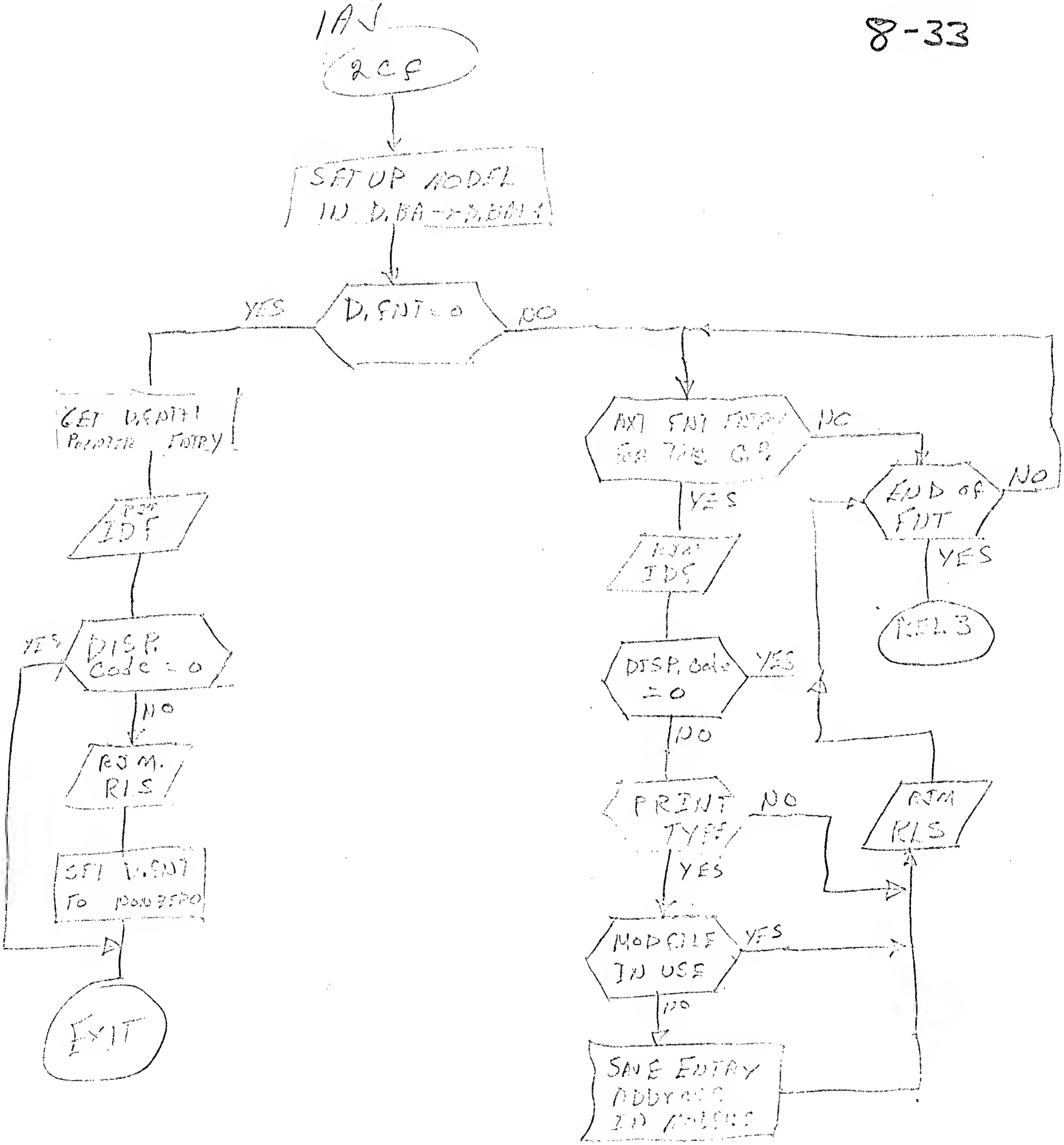
yes
NO
Send
Lock
Create
IS there a
Good Check
Return To
Save Address

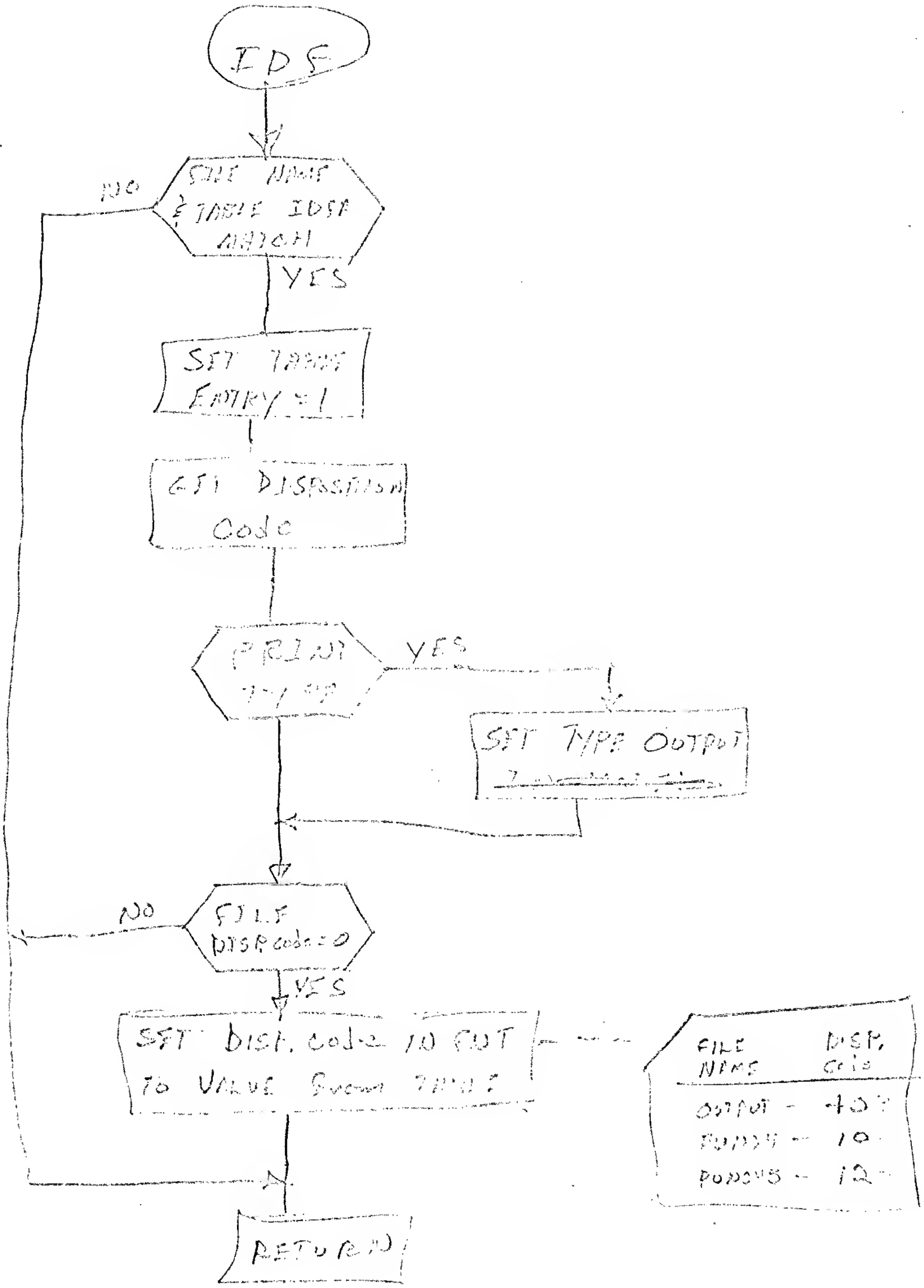

Is next Entry Job file
yes
NO
Is it assigned Link file
yes
NO
Is it output file
NO
yes
Save output file FNT Address
Look FNT entry
Increment Index
End of stack
NO
yes
Request storage for Buffer
Request Honored
NO
Send Dayfile message waiting for storage
Drop PP
IDLE Loop
yes
Reserve FNT Pseudo Channel
Look for FNT Entry space
Space Available
NO
Send Dayfile message FNT/FST table Jammed
Release Pseudo Channel
Drop PP
IDLE Loop
yes
1


1AJ/2CJ
NO
YES
READ
INTO CM
OUTPUT CM
RESET D.F.B.
AND FET
RETURN

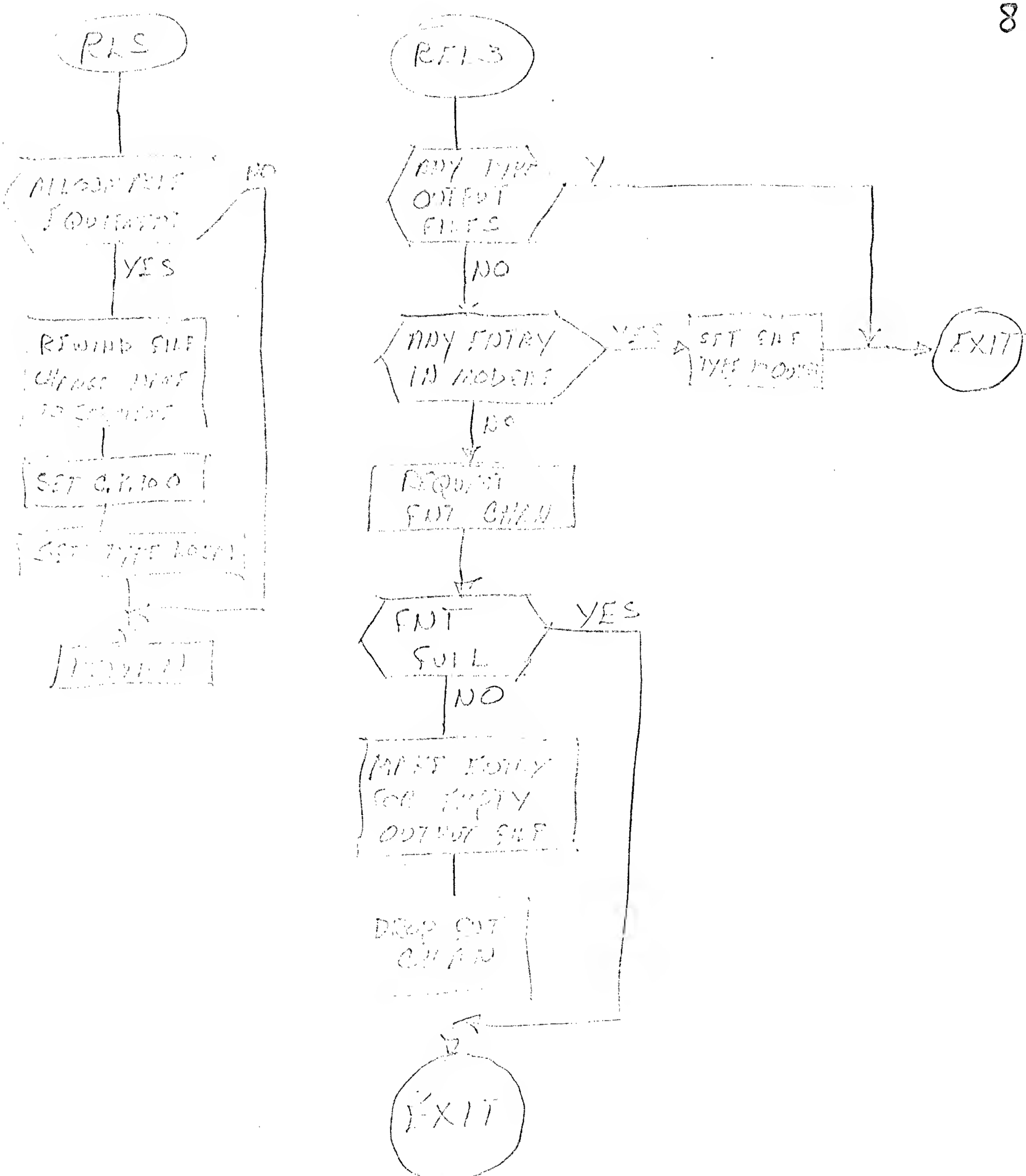


1AJ
YES
NO
ANY FILE
YES
WRITE EOR
CALL DMP
PP
DMP
COMPLETE
NO
TO 1AJ

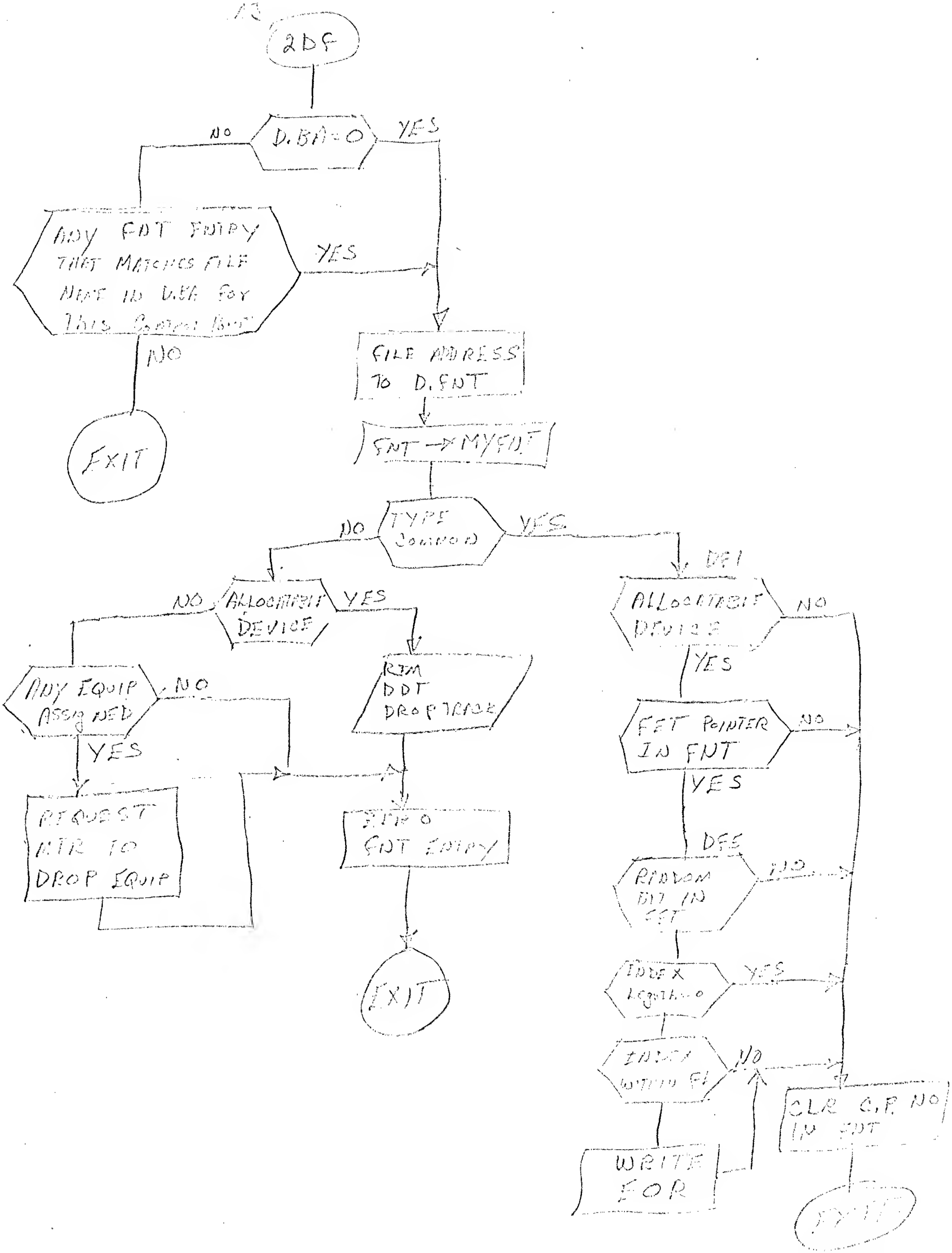

1AJ
2CF
SET UP MODEL IN D.BA
D.FNT = 0
YES
NO
GET DEPTH POINTER ENTRY
IDF
DISP. Code = 0
YES
NO
RJM. RLS
SET D.FNT TO NONZERO
EXIT
NXT FNT ENTRY FOR THE C.P.
NO
YES
IDF
DISP. Code = 0
YES
NO
PRINT TYPE
NO
YES
MODFILE IN USE
YES
NO
SAVE ENTRY ADDRESS
END OF FNT
NO
YES
RJM RLS

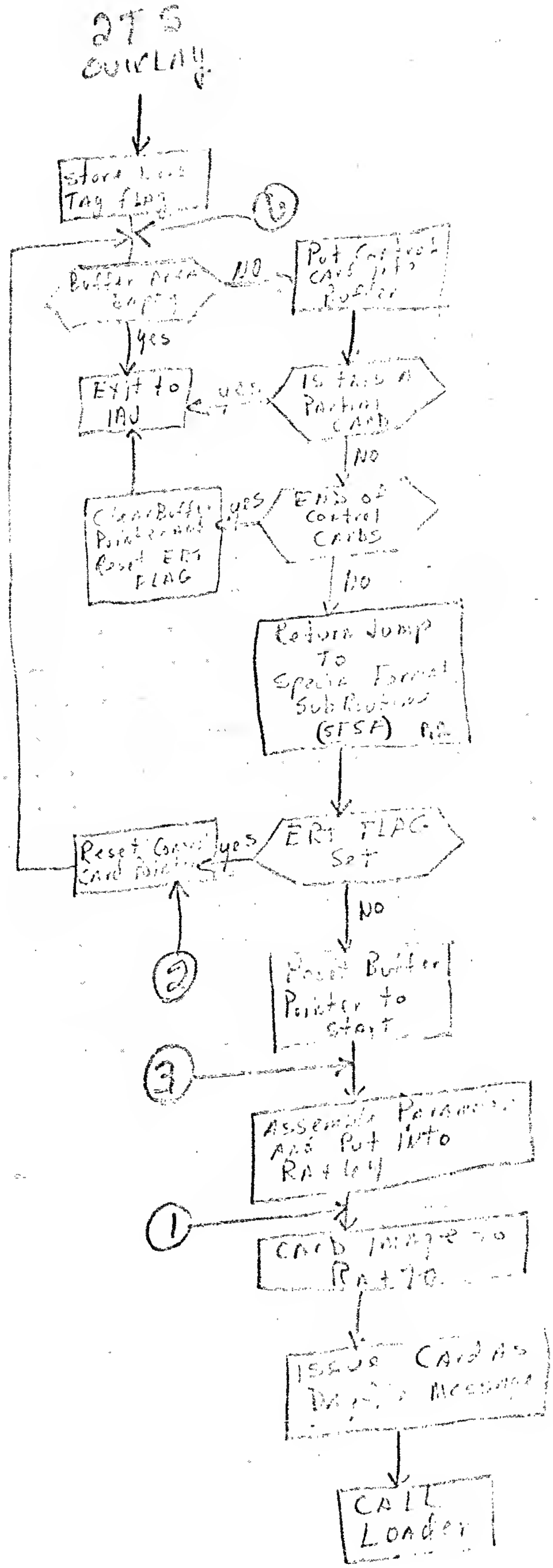

IDF
FILE NAME & TABLE IDSP MATCH
NO
YES
SET TABLE ENTRY = 1
GET DISPOSITION Code
PRINT
YES
SET TYPE OUTPUT
FILE DISP code = 0
NO
YES
SET DISP. code IN FET TO VALUE FROM TABLE
FILE NAME
DISP. Code
OUTPUT - 40
PUNCH - 10
PUNCHB - 12
RETURN

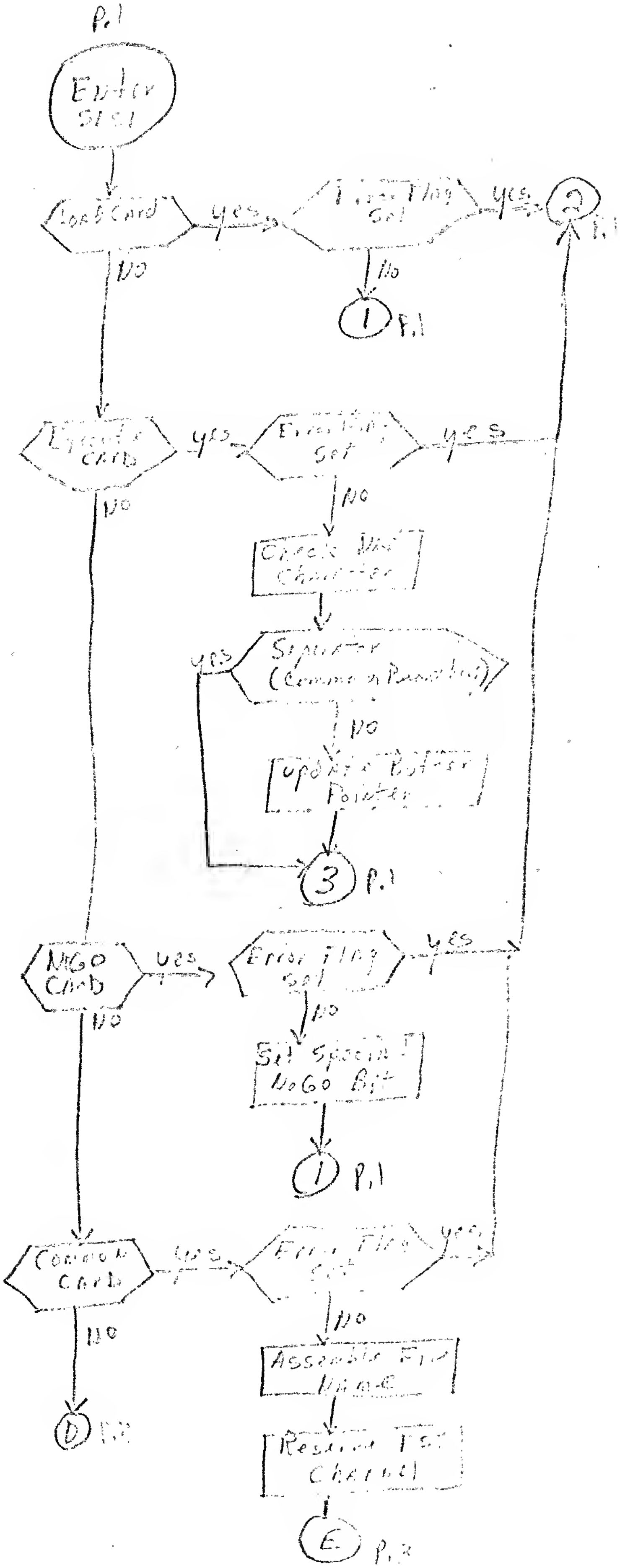
3/3
8-35



RLS
NO
YES
REWIND FILE
REL3
Y
NO
YES
EXIT
NO
FNT
FULL
YES
NO
EXIT


2DF
D.BA=0
NO
YES
ANY FNT ENTRY THAT MATCHES FILE
YES
NO
EXIT
FILE ADDRESS TO D.FNT
FNT → MYFNT
TYPE COMMON
NO
YES
DFI
ALLOCATABLE DEVICE
NO
YES
ALLOCATABLE DEVICE
NO
YES
ANY EQUIP ASSIGNED
NO
YES
REQUEST MTR TO DROP EQUIP
RTM DDT DROP TRACE
FNT ENTRY
EXIT
FET POINTER IN FNT
NO
YES
DFS
RANDOM BIT IN FET
NO
INDEX
YES
INDEX WITHIN FL
NO
WRITE EOR
CLR C.P. NO IN FNT


2TS
OVERLAY
Store ... Tag flag
Buffer Area Empty
NO
yes
Put Control Card into Buffer
Exit to IAJ
yes
Is this a Partial Card
NO
Clear Buffer Pointer and Reset ERT FLAG
yes
End of Control Cards
NO
Return Jump To Special Format SubRoutine (SFSR)
Reset Control Card Pointer
yes
ERT FLAG Set
NO
2
Reset Buffer Pointer to Start
3
Assemble Parameters And Put Into RA+164
1
Card Image to RA+70
Issue Card As Display Message
CALL Loader



P.1
Enter
Load Card
yes
Error Flag Set
2
No
1 P.1
Execute Card
yes
Error Flag Set
Check Next Character
Separator (Comma or Parenthesis)
Update Buffer Pointer
3 P.1
NOGO Card
Set Special NoGo Bit
Common Card
Assemble File Name
Restore Tst Channel
D
E

PRESET STORAGE SUBROUTINE   10T TRANSIENT PROGRAM   SCOPE 3

PAGE 1 → CLEAR RA THRU RA+77₈ → SET UP BUFFER PARAMETERS FOR EACH OUTPUT DEVICE → SET THE "STORAGE SET" FLAG (0001 IN BYTE 4 OF I.R.) → STORE HEADINGS FOR BUFFER POINTS IN PRIMARY MSG BUFFER AT CP AREA (CPA address 30-34) 1. IDLE 2. IDLE 3. IDLE --- 6. IDLE → RETURN TO MAIN PROGRAM

see also p. 8-17

(IF ONLY 3 OUTPUT DEVICES, THEN SET UP ONLY 3, ETC.)

RA+11 = 0——0100 ⇒ FIRST
RA+12 = 0——0100 ⇒ IN
RA+13 = 0——0100 ⇒ OUT
RA+14 = 0——01100 ⇒ LIMIT
} OUTPUT DEVICE #1 AT BUFFER POINT #1

RA+21 = 0——01100 ⇒ FIRST
RA+22 = 0——01100 ⇒ IN
RA+23 = 0——01100 ⇒ OUT
RA+24 = 0——02100 ⇒ LIMIT
} OUTPUT DEVICE #2 AT BUFFER POINT #2

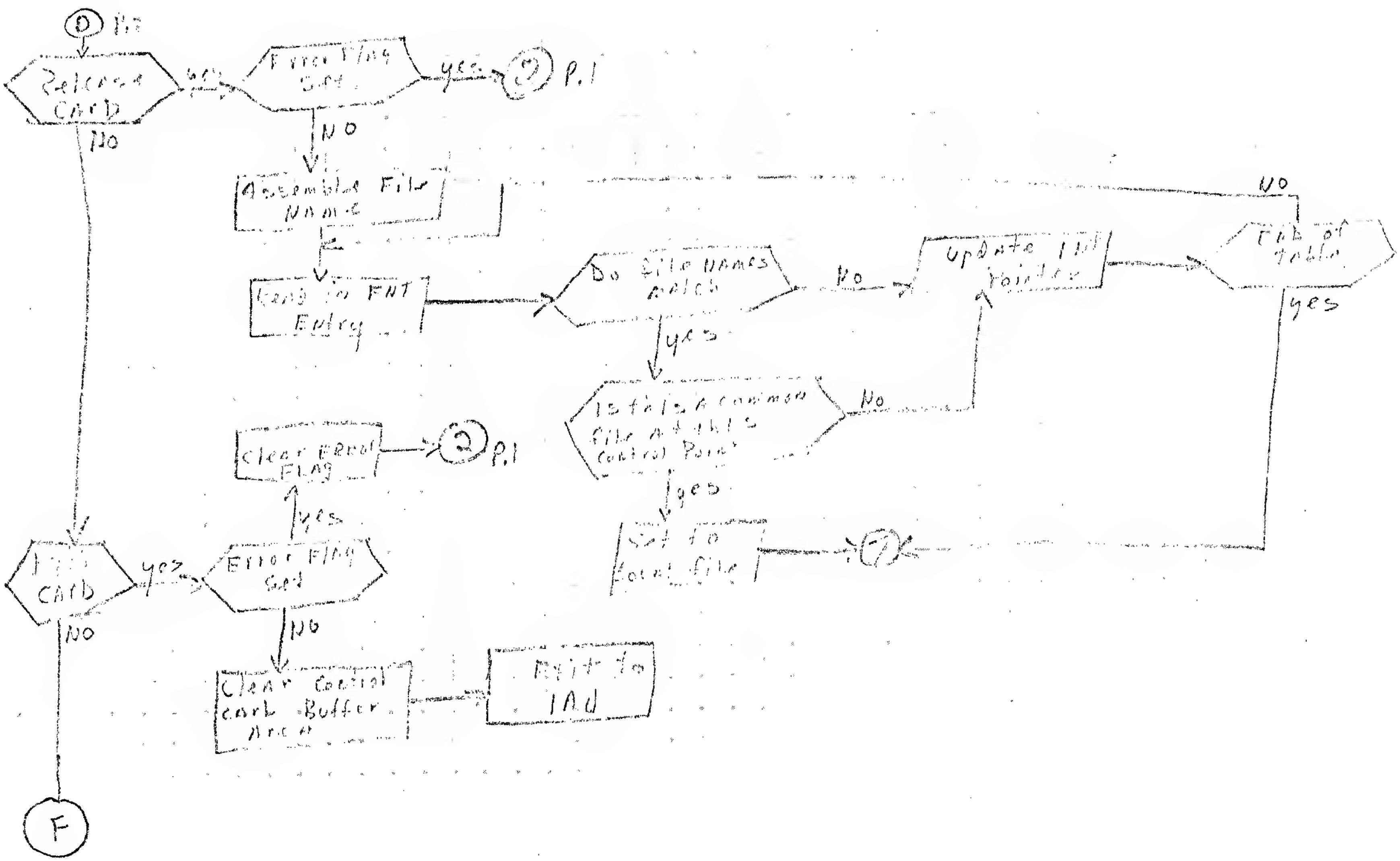
#3

#4

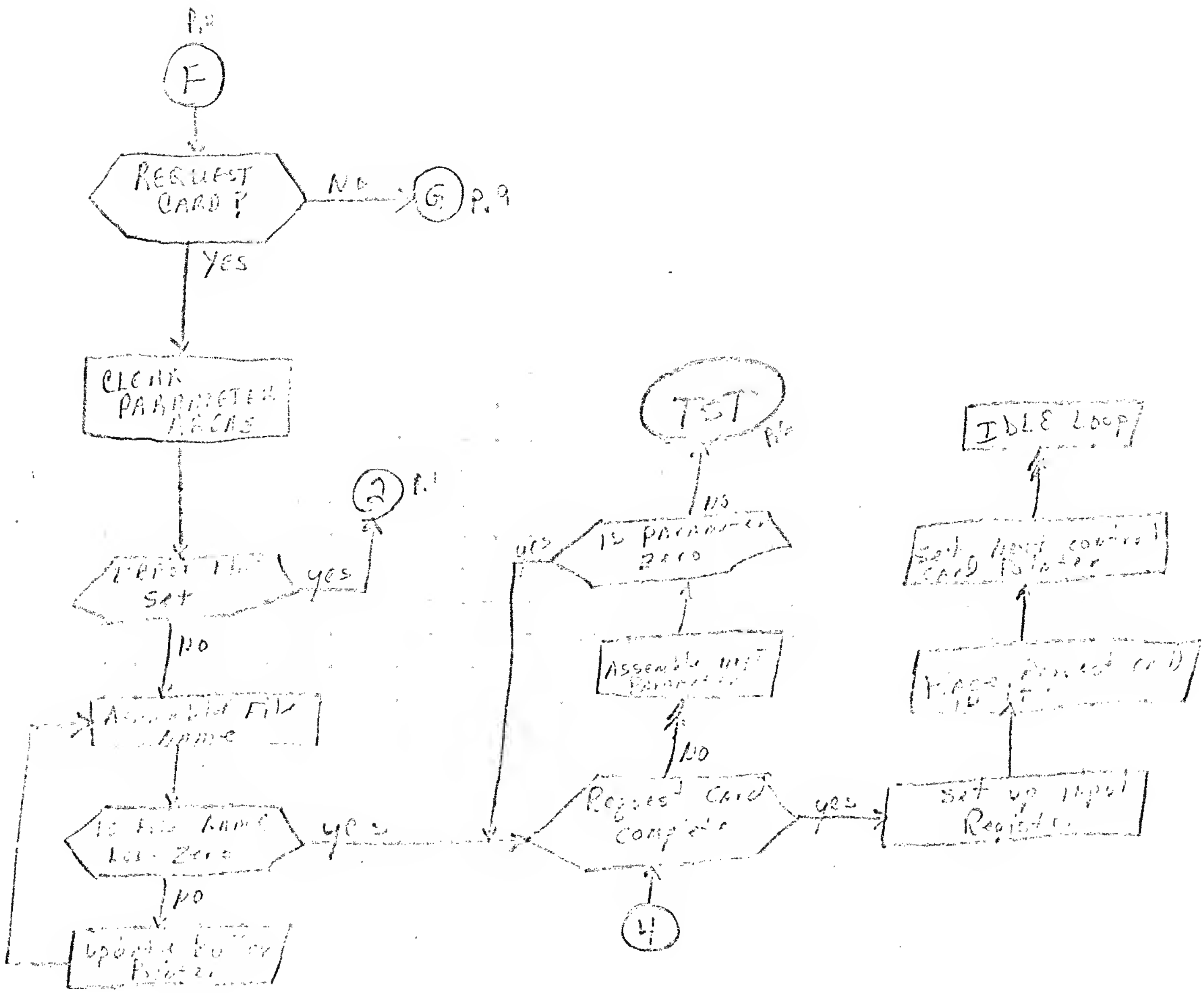
#5

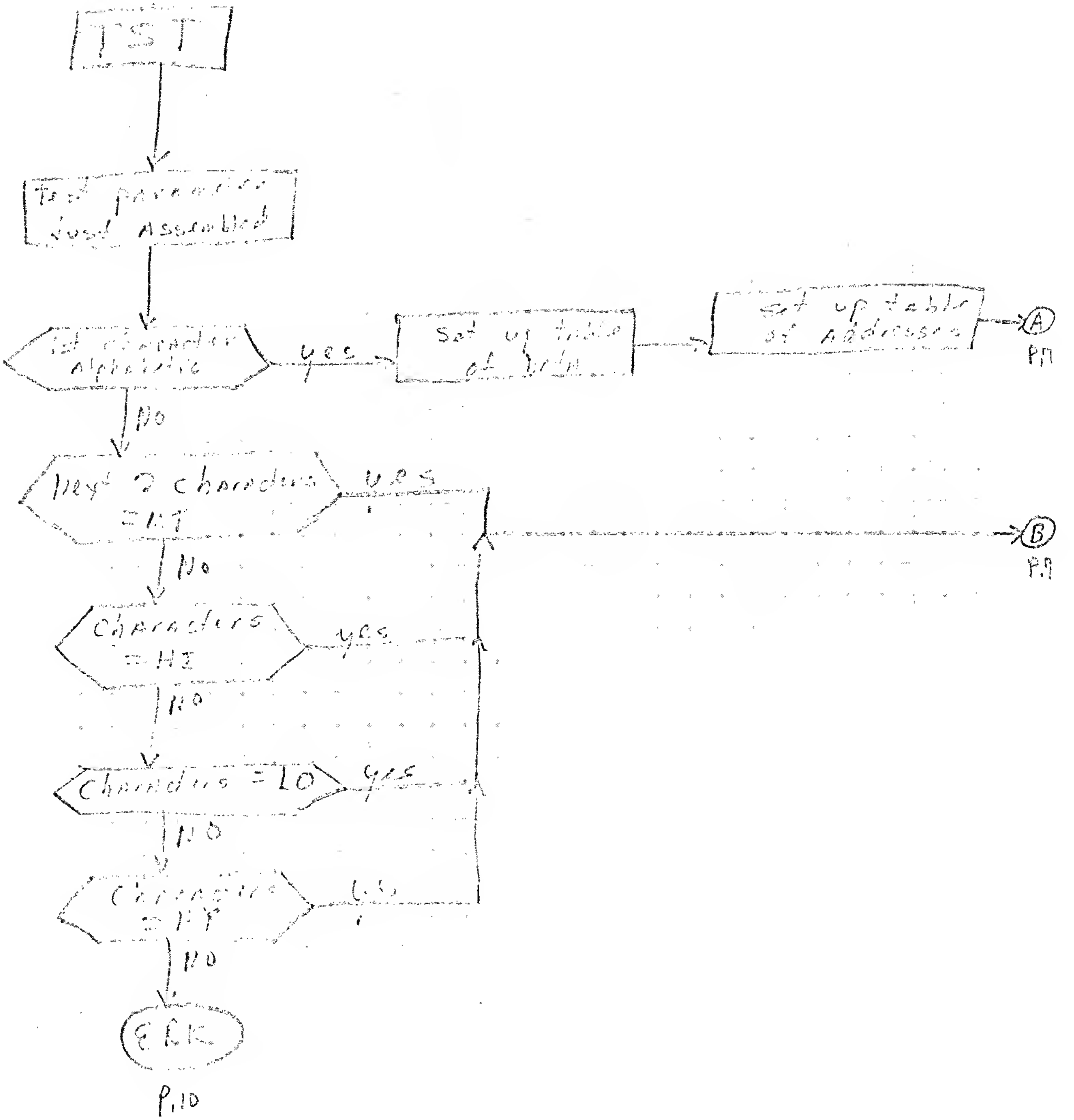
RA+61 = 0——05100 ⇒ FIRST
RA+62 = 0——05100 ⇒ IN
RA+63 = 0——05100 ⇒ OUT
RA+64 = 0——06100 ⇒ LIMIT
} OUTPUT DEVICE #6 AT BUFFER POINT #6

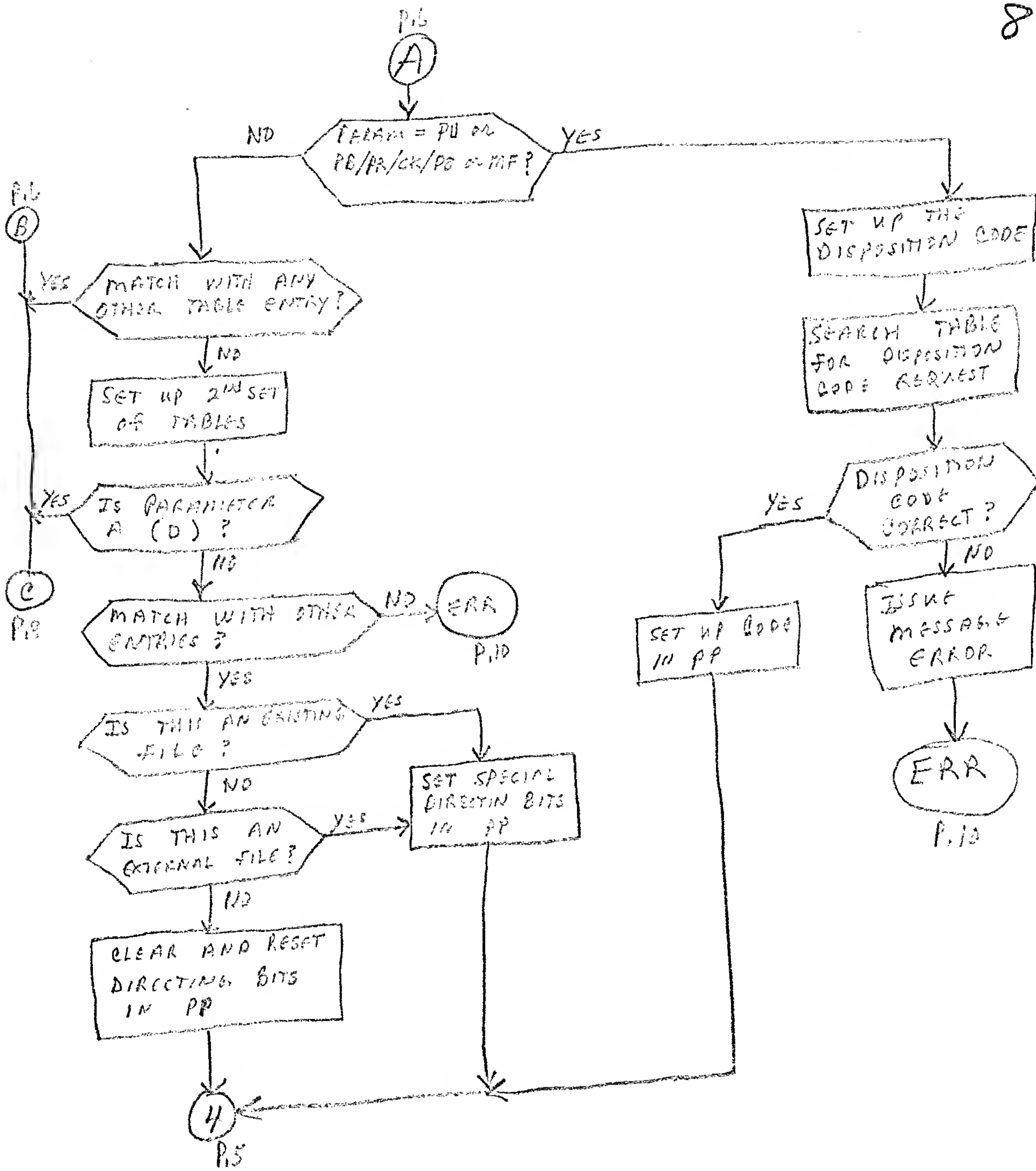
8-34

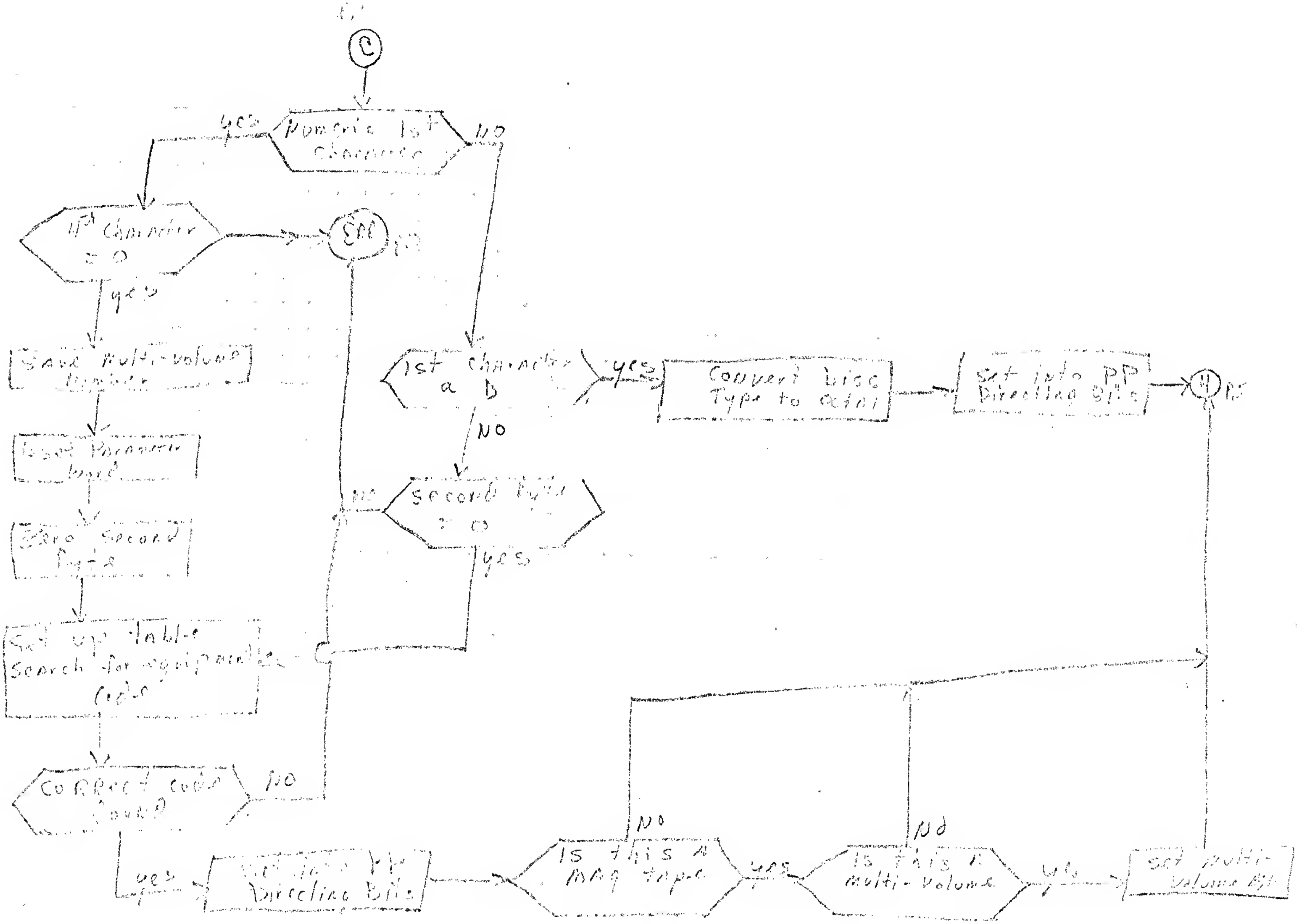

Release CARD
Error Flag Set
P.1
Assemble File Name
Read in FNT Entry
Do file names match
Update FNT Pointer
End of table
Is this a common file at this control point
Set to local file
Clear Error FLAG
P.1
CARD
Error Flag Set
Clear Control card Buffer Area
Exit to
F

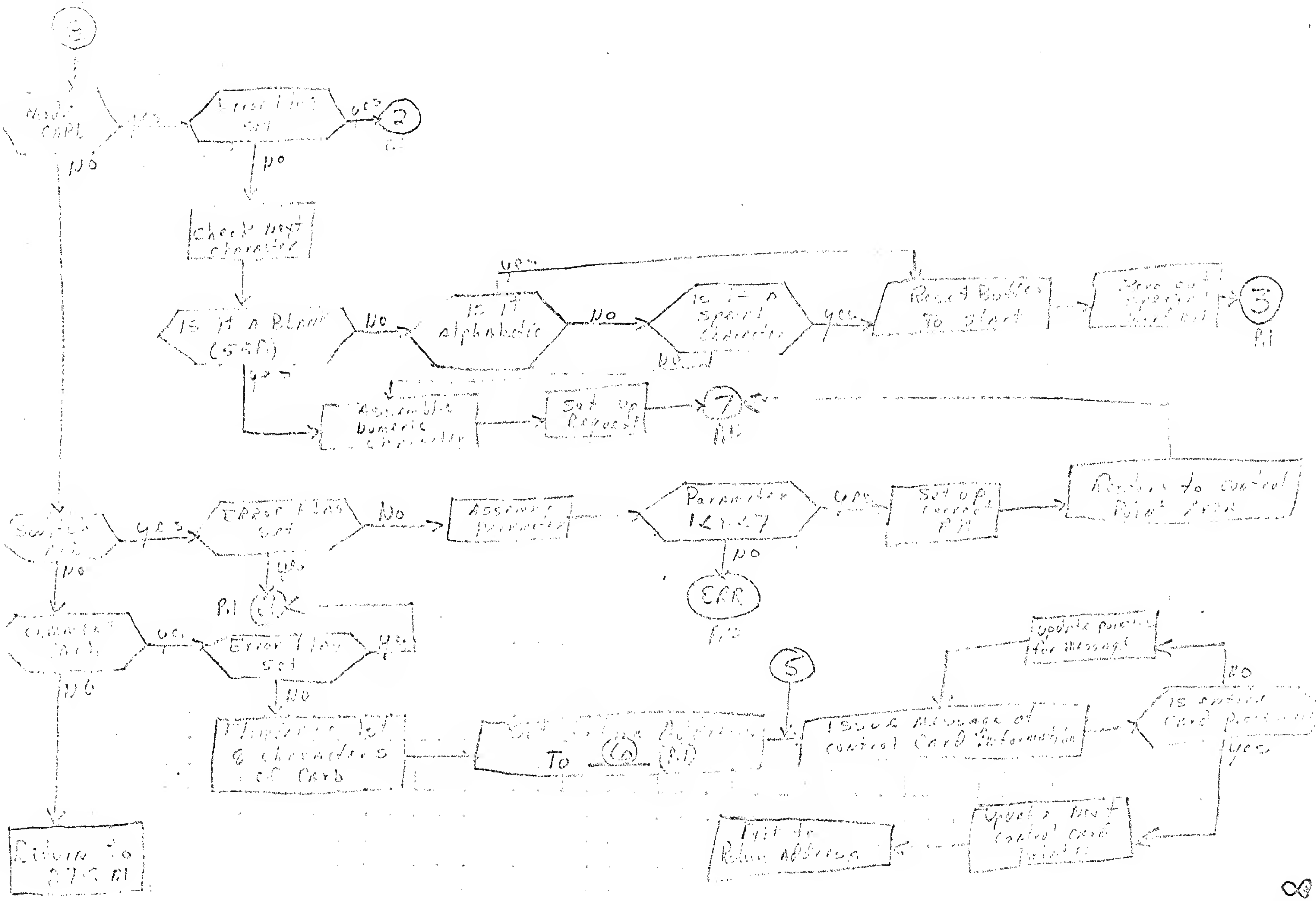
8-40


P.8
F
REQUEST CARD?
NO
G
P.9
YES
CLEAR PARAMETER AREAS
2
yes
no
Assemble File Name
Is File Name
no
Update Buffer Pointer
yes
TST
no
Is Parameter Zero
yes
Assemble next Parameter
no
Request Card Complete
yes
4
Set up Input Registers
Set Next control Card Pointer
IDLE Loop

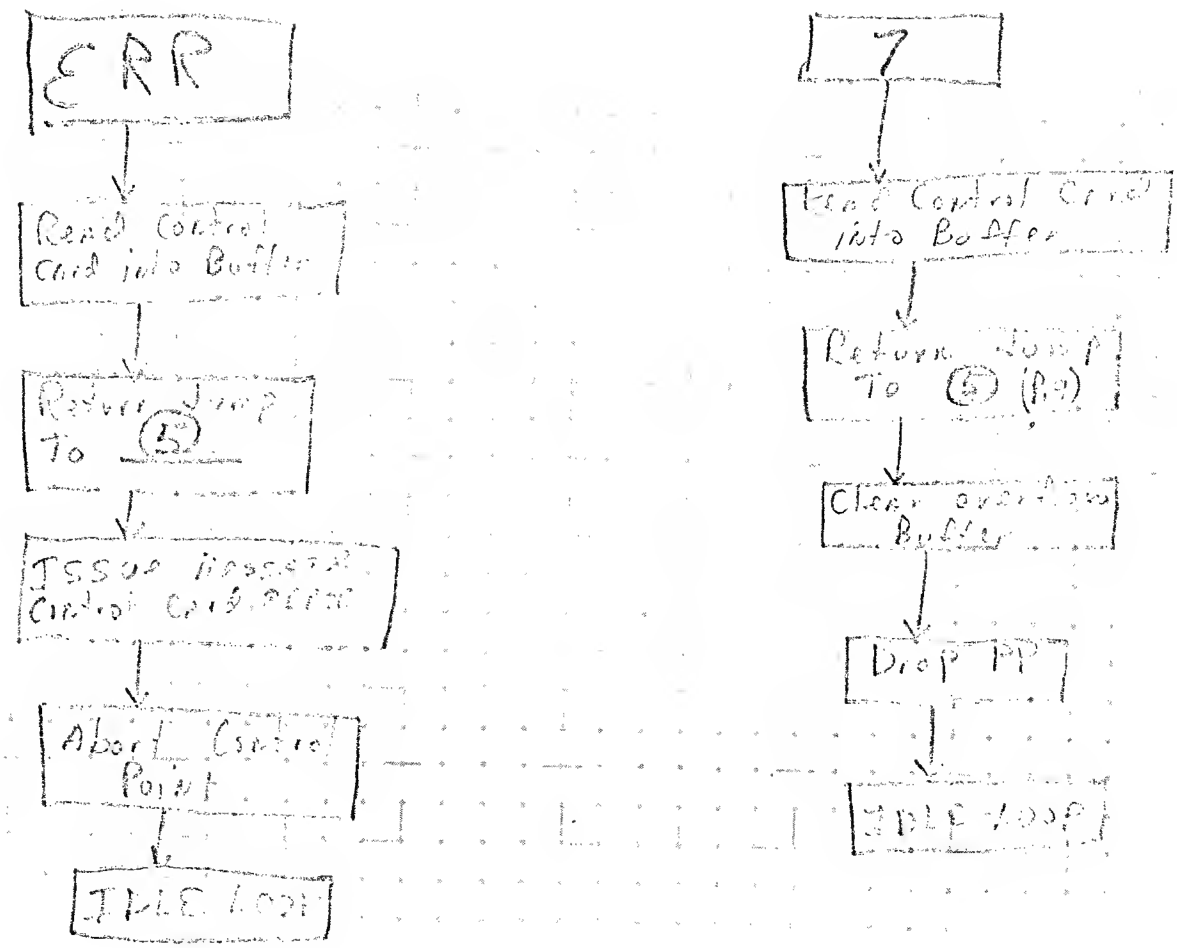

TST
Test parameter just assembled
1st character alphabetic
yes
Set up table of data
Set up table of addresses
A
P.11
NO
Next 2 characters = AT
yes
B
P.11
NO
Characters = HI
yes
NO
Characters = LO
yes
NO
Characters = FY
yes
NO
ERR
P.10


P.6
A
PARAM = PU or PB/PR/CK/PE or MF?
NO
YES
SET UP THE DISPOSITION CODE
SEARCH TABLE FOR DISPOSITION CODE REQUEST
DISPOSITION CODE CORRECT?
YES
NO
ISSUE MESSAGE ERROR
ERR
P.10
SET UP CODE IN PP
P.6
B
MATCH WITH ANY OTHER TABLE ENTRY?
YES
NO
SET UP 2ND SET OF TABLES
IS PARAMETER A (D)?
YES
NO
C
P.8
MATCH WITH OTHER ENTRIES?
NO
ERR
P.10
YES
IS THIS AN EXISTING FILE?
YES
NO
SET SPECIAL DIRECTIN BITS IN PP
IS THIS AN EXTERNAL FILE?
YES
NO
CLEAR AND RESET DIRECTING BITS IN PP
4
P.5

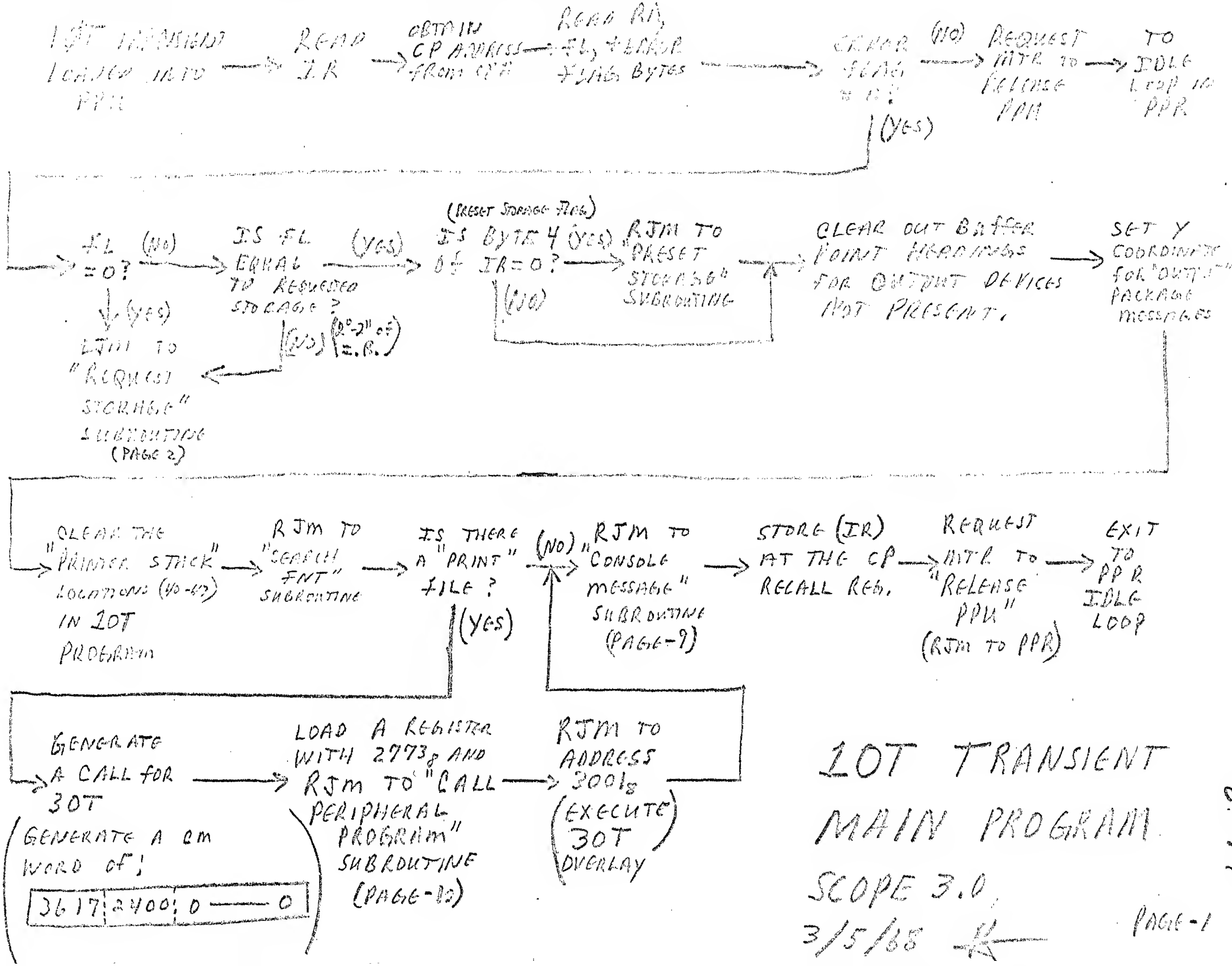

C
Numeric 1st Character
yes
NO
4th Character = 0
EM
yes
Save multi-volume number
Zero Second Byte
Set up table search for equipment code
Correct code found
NO
yes
Set into PP Directing Bits
Is this A MAG tape
NO
yes
Is this A multi-volume
NO
yes
Set multi-volume Bit
1st Character a D
yes
NO
Convert Disc Type to octal
Set into PP Directing Bits
second Byte = 0
NO
yes



NO
yes
no
Check next character
yes
Is it a Blank (55B)
NO
Is it alphabetic
No
Is it a Special character
yes
Reset Buffer to start
P.1
Assemble Numeric character
Set up Repeat
7
yes
NO
Error Flag set
No
Assemble parameter
Parameter
yes
Set up correct
NO
ERR
P.1
yes
Error Flag set
No
yes
5
Issue Message of control Card Information
Update pointer for Message
NO
yes
To
Exit to Return Address


ERR
Read Control Card into Buffer
Return Jump To 5
Issue
Control Card
Abort Control Point
IDLE LOOP
7
Read Control Card into Buffer
Return Jump To 5
Clear overflow Buffer
Drop PP
IDLE LOOP

1OT MAIN PROGRAM

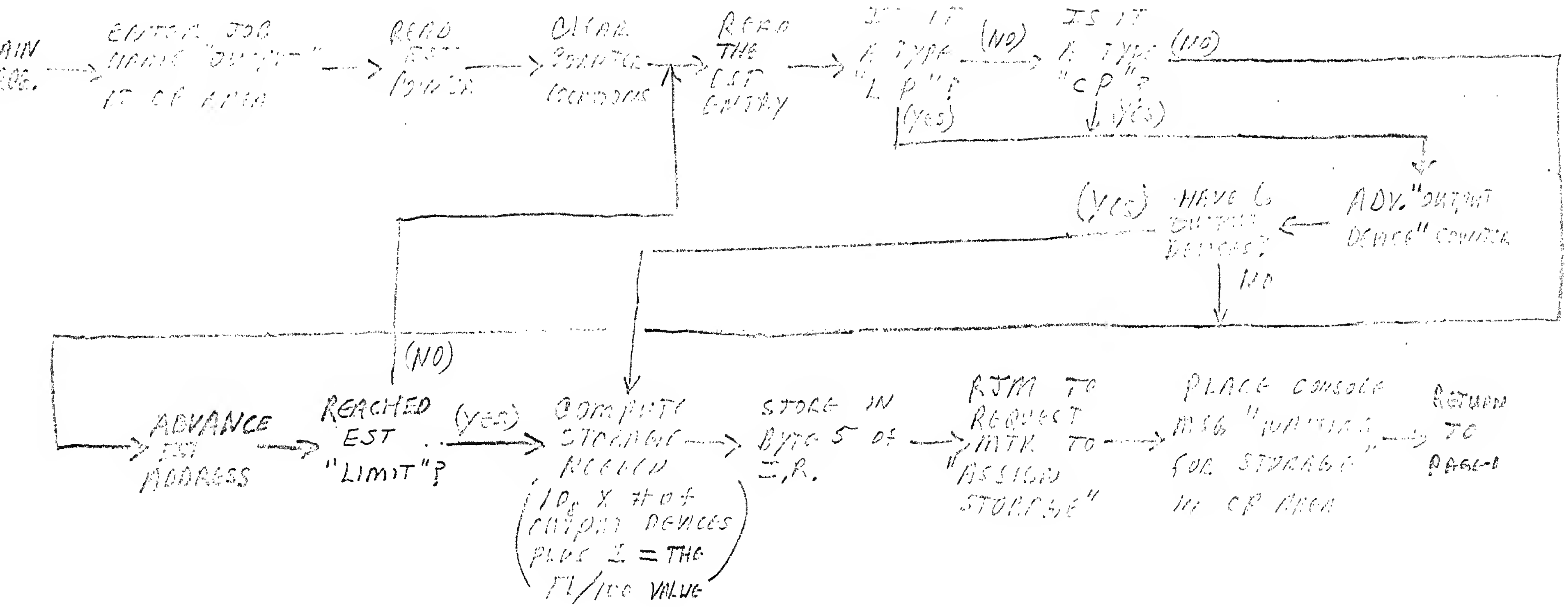


1OT TRANSIENT MAIN PROGRAM

SCOPE 3.0

3/5/65

REQUEST STORAGE SUBROUTINE



B4-8

PRESET STORAGE SUBROUTINE — 10T TRANSIENT PROGRAM — SCOPE 3

PAGE 2 → CLEAR RA THRU RA+77$_8$ → SET UP BUFFER PARAMETERS FOR EACH OUTPUT DEVICE → SET THE "STORAGE SET" FLAG (0001 IN BYTE 4 OF I.R.) → STORE HEADINGS FOR BUFFER POINTS IN PRIMARY MSG BUFFER AT CP AREA → RETURN TO MAIN PROGRAM

(IF ONLY 3 OUTPUT DEVICES, THEN SET UP ONLY 3, ETC.)

RA+11 = 0——0100 ⇒ FIRST
RA+12 = 0——0100 ⇒ IN
RA+13 = 0——0100 ⇒ OUT
RA+14 = 0——01100 ⇒ LIMIT
} OUTPUT DEVICE #1 AT BUFFER POINT #1

RA+21 = 0——01100 ⇒ FIRST
RA+22 = 0——01100 ⇒ IN
RA+23 = 0——01100 ⇒ OUT
RA+24 = 0——02100 ⇒ LIMIT
} OUTPUT DEVICE #2 AT BUFFER POINT #2

#3

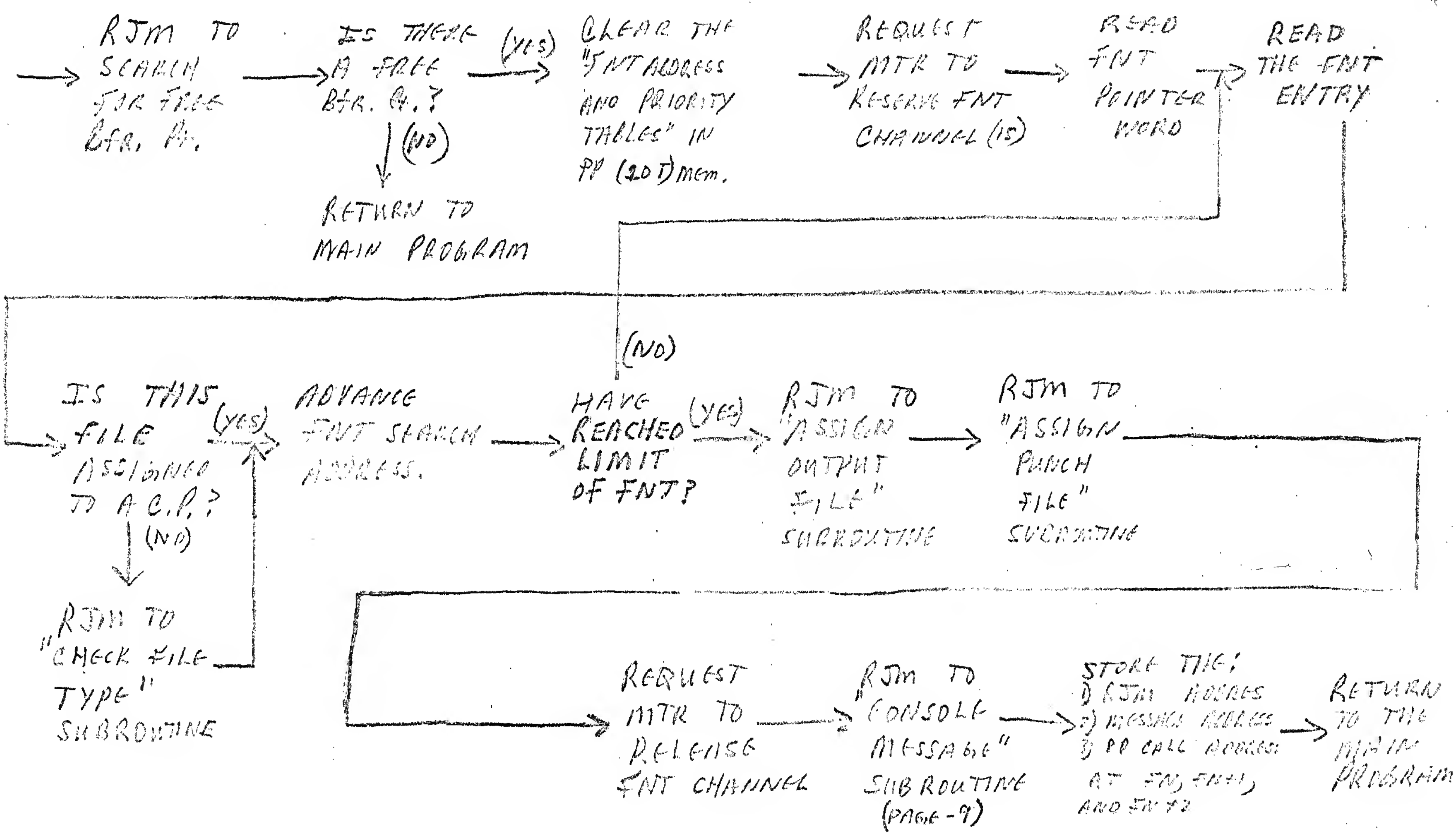
#4

#5

RA+61 = 0——05100 ⇒ FIRST
RA+62 = 0——05100 ⇒ IN
RA+63 = 0——05100 ⇒ OUT
RA+64 = 0——06100 ⇒ LIMIT
} OUTPUT DEVICE #6 AT BUFFER POINT #6

617-8

SEARCH FNT SUBROUTINE 20T TRANSIENT PROGRAM, SCOPE 3



29-8

SEARCH FOR FREE BUFFER SUBROUTINE — 1OT TRANSIENT PROGRAM, SCOPE 3

START ... → READ PRESENT "BUFFER NAME WORD" FOR BUFFER POINT # 1 (RA+15), 2 (RA+25), 3, 4, 5, 6 (RA+65) → BYTE 1 OF WORD = 0 ?

(NO) → ADVANCE "BFR. PT. # FREE" ADDRESS → SEARCHED ALL THE BFR. PT.'S ? (NO) → back to READ; (YES) → CLEAR THE "BUFFER POINT # FREE" ADDRESS → RETURN TO PREVIOUS PROGRAM

(YES) → CLEAR TIME ACCUMULATION WORD (RA+17, BFR. PT. #1) (RA+27, BFR. PT. #2) ETC. (RA+67, BFR. PT #6) → RETURN TO PREVIOUS PROGRAM

NOTE 1: BFR. PT. # FREE ADDRESS CONTAINS THE BFR. PT. # THAT IS FREE. A "0" ⇒ NONE ARE FREE.

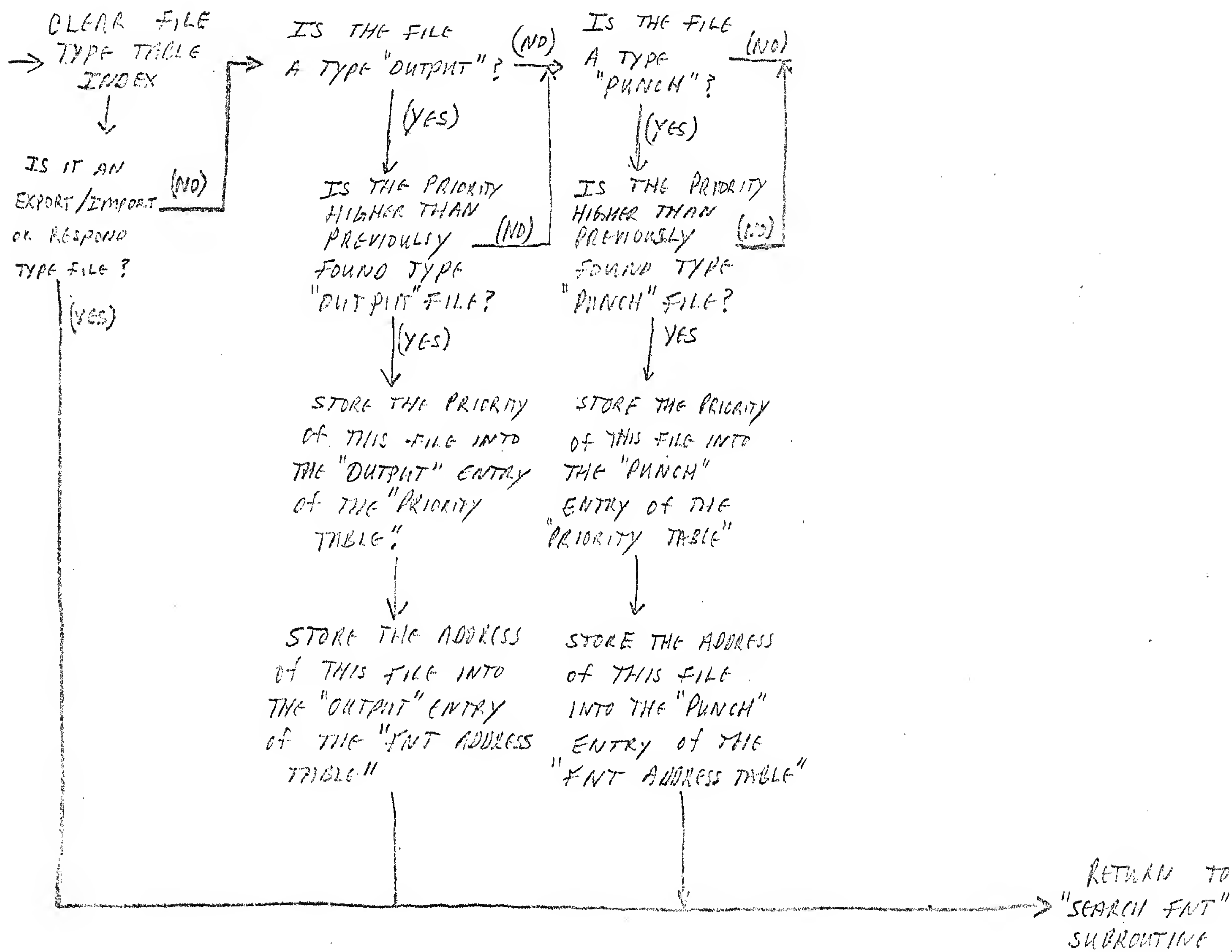
REQUEST EQUIPMENT SUBROUTINE — 1OT TRANSIENT PROGRAM, SCOPE 2.0, PSR # 47

4/20/67

ENTER SUBROUTINE WITH (A) EQUAL TO TYPE OF EQUIPMENT DESIRED IN DPC. → RJM TO REQUEST MTR TO "ASSIGN EQUIP". (TO PPR) → READ 1st WORD OF MESSAGE BUFFER FOR THIS PPU. → LOAD THE SYSTEM EQUIPMENT # THE MTR ASSIGNED IN A REG. → RETURN TO SUBROUTINE

19-8

CHECK FILE TYPE SUBROUTINE — 1DT TRANSIENT PROGRAM



8-52

ASSIGN OUTPUT FILE SUBROUTINE 1DT TRANSIENT PROGRAM, SCOPE 3
(ASSIGN "PRINT" FILE)

IS THERE AN FNT ADDRESS AT THE "OUTPUT" ENTRY?

(YES) → IS THERE A BFR. PT. # FREE?

(NO) → RETURN TO "SEARCH FNT" SUBROUTINE

(YES) → RJM TO "REQUEST EQUIPMENT" SUBROUTINE (PAGE-5)

(NO) → CLEAR OUT THE "OUTPUT" ENTRY IN THE FNT ADDRESS TABLE → RETURN TO "SEARCH FNT" SUBROUTINE

→ DID MTR. ASSIGN AN "LP"?

(NO) → CLEAR OUT THE "OUTPUT" ENTRY IN THE FNT ADDRESS TABLE

(YES) → STORE THE SYSTEM # OF THE LP ASSIGNED BY MTR IN THE "PRINTER STACK"
(PDK MEMORY LOCATIONS;
41 = BFR. PT. #1
42 = " " #2
46 = BFR. PT. #6)

→ READ THE EST POINTER WORD → ADD SYSTEM # OF LP ASSIGNED TO POINTER → READ EST ENTRY FOR ASSIGNED LP → SET BYTE 1 = 3004₈ FOR PRINTER ENTRY POINT → BFR. PT. # "X" IN BYTE 4 "00X0" → STORE EST ENTRY WITH THESE CHANGES AT RA + X6 (X = BFR. PT.)

3004	0000	0000	00X0	PRINTER CODES

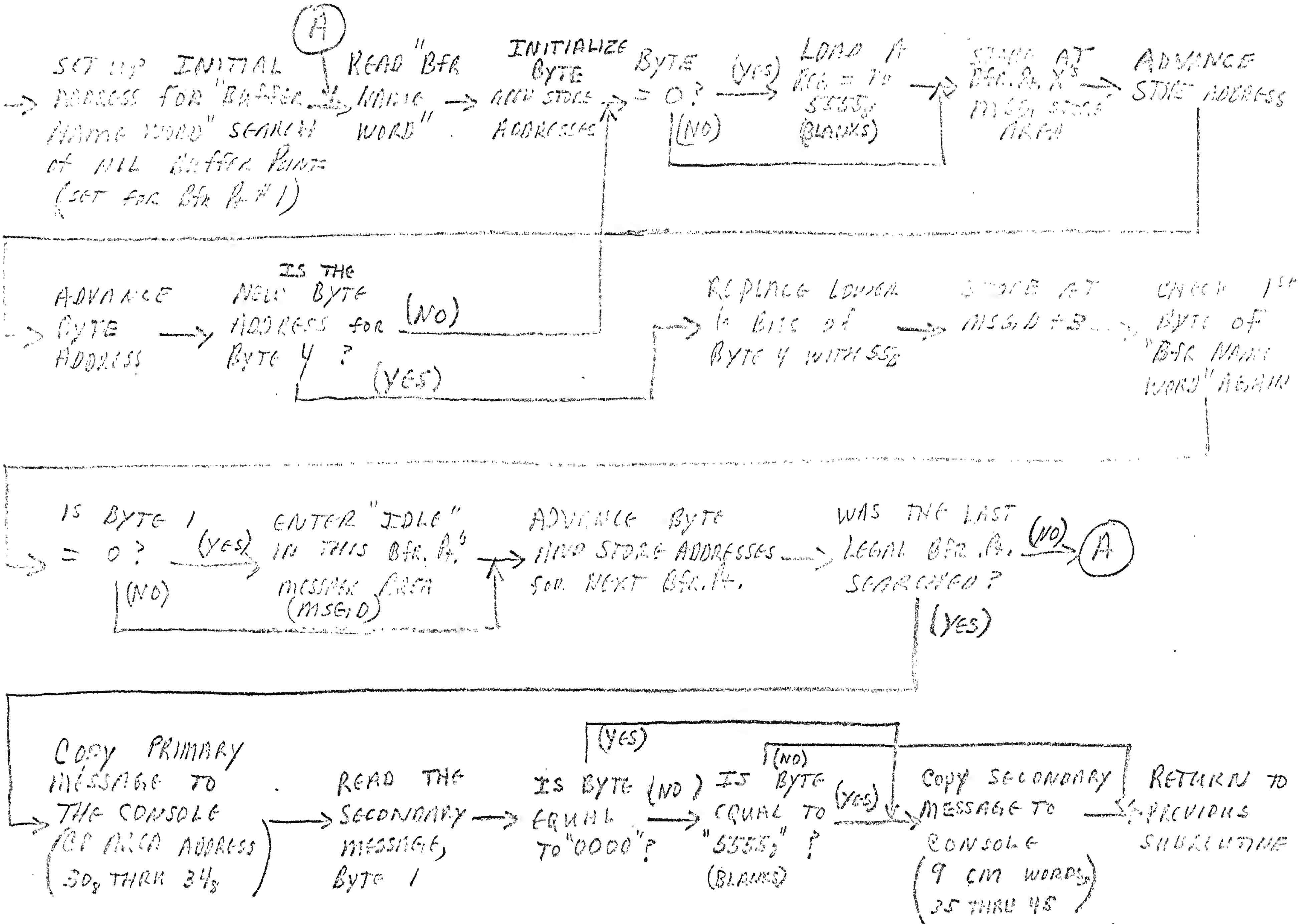
→ GENERATE "PRINT" IN DPC AND STORE AT CONSOLE INFO. AREA.
(2164-2167 FOR BFR PT #1
2170-2173 " " " #2
2174-2177 " " " #3
2200-2203 " " " #4
2204-2207 " " " #5
2210-2213 " " " #6)

→ RJM TO "ASSIGN FILE" SUBROUTINE
((A) = FNT ADDRESS OF OUTPUT FILE)
PAGE-8

→ RETURN TO "SEARCH FNT" SUBROUTINE (PAGE-4)

8-53

CONSOLE MESSAGE SUBROUTINE | IOT TRANSITION PROGRAM, SCOPE 3



8-59

PAGE-9

CALL PERIPHERAL PROGRAM — 10T TRANSIENT PROGRAM, SCOPE 3

ON ENTRY, THE A REGISTER EQUALS ADDRESS OF 1ST BYTE OF NAME OF PROGRAM TO BE CALLED

→ COPY NAME OF PROGRAM DESIRED TO 1ST WORD OF MESSAGE BUFFER FOR THIS PPU

→ REQUEST MTR TO "ASSIGN A PPU" TO EXECUTE THIS PROGRAM

→ PAUSE FOR STORAGE RELOCATION

→ CHECK BYTE 1 OF MSG. BFR REG. TO SEE IF MTR HAD ASSIGNED A PPU, BYTE EQUAL "0"?

(YES) (NOTE – BYTE 1 SHOULD = NEW PPU INPUT REG. ADDRESS IF ASSIGNED)

(NO) → RETURN TO "ASSIGN PUNCH FILE" SUBROUTINE

8-57
1/3

30T

30T

·PRS·

insert CP # in CIO calls

zero cycle count

A →

Buffer empty? $\xrightarrow{NO}$ LPD4

YES

FET status contain End-of-Information? $\xrightarrow{YES}$ ·PTIM· TIME ACCOUNTING → LIM to TER

1st time for this file? $\xrightarrow{NO}$ set OUT = FIRST + IN = FIRST+5

YES

call CIO

To READ DISK FILE

LPC

Try to change to another printer + file

Another LP & FILE READY·BUSY → A

LP04

at least 5 words in buffer? → NO → CIO to get more

YES

STS to get LP status. status = 0? → NO → LPC to switch printers & files

setup spacing format

STS · printer Ready · $\overline{\text{BUSY}}$ → NO

YES

convert DISPLAY code to BCD

call PRT to Print (# of Bytes in WC)

ADD PRINT TIME for accounting

R

8-57
3/3

TER

is REPEAT flag set at LP+0 ? NO → get EST ordinal for LP → call CLP in 1OT TO DROP PRINTER

·LPC·
check for active LP.

YES

clr flag.
wait for File Busy
clr EOI flag in FET

CALL CIO to Rewind file

A

LPC

store return addr. in FET+6

clr PR
increase CC by 1

CC=100 NO → ADD 1 to PS (which FET we are concerned with) if its too high set PS= 1.

YES

call FNT in 1OT to find new print file

is (PS+FET#) = zero → increase PR # by 1 to find a new printer → is PR > total number of FET's

NO

Read new FET → PP

set return addr. from FET+6 → are bits 0-11 of RA+0=0 YES → EXIT to LPD cycle for new FET

NO

PROCESS FLAG

YES

all FET'S have been checked for an active LP.
EXIT 3OT
BACK TO 1OT

6000 SCOPE LOADER

(SEE ALSO APPENDIX D of SCOPE 3.1 Reference Manual)

Address							
RA	ERROR EXIT.						
RA+1	COMMUNICATIONS						
RA+2 … RA+63	PARAMETERS FROM THE "PROGRAM CALL" CARD UPON AVAILABLE TO USER DURING EXECUTION FOR FILE NAMES AND OTHER USAGE						
RA+64	PROGRAM CALL OR FILE NAME						NUMBER OF PARAMETERS
RA+65		FWA SEGMENT TABLE	35	28	24	18	CORE NEXT
RA+66		FWA LOADER TABLES	DEBUG CONTROL BITS(31-28)		U	V	FWA PROGRAM
RA+67		W		X	Y		FWA LOADER
RA+70 … RA+77	CARD IMAGE OF CARD WHICH CALLED FOR EXECUTION						

U (4 BITS) LDR DIRECTIVES

V (6 BITS) LEVEL OF INCOMING OVERLAY

W SEGMENT LOAD - ADDRESS OF LOWEST LINKAGE TABLE

OVERLAY LOAD - ADDRESS OF BLANK COMMON

X (5 BITS) | E | D | C | B | A |

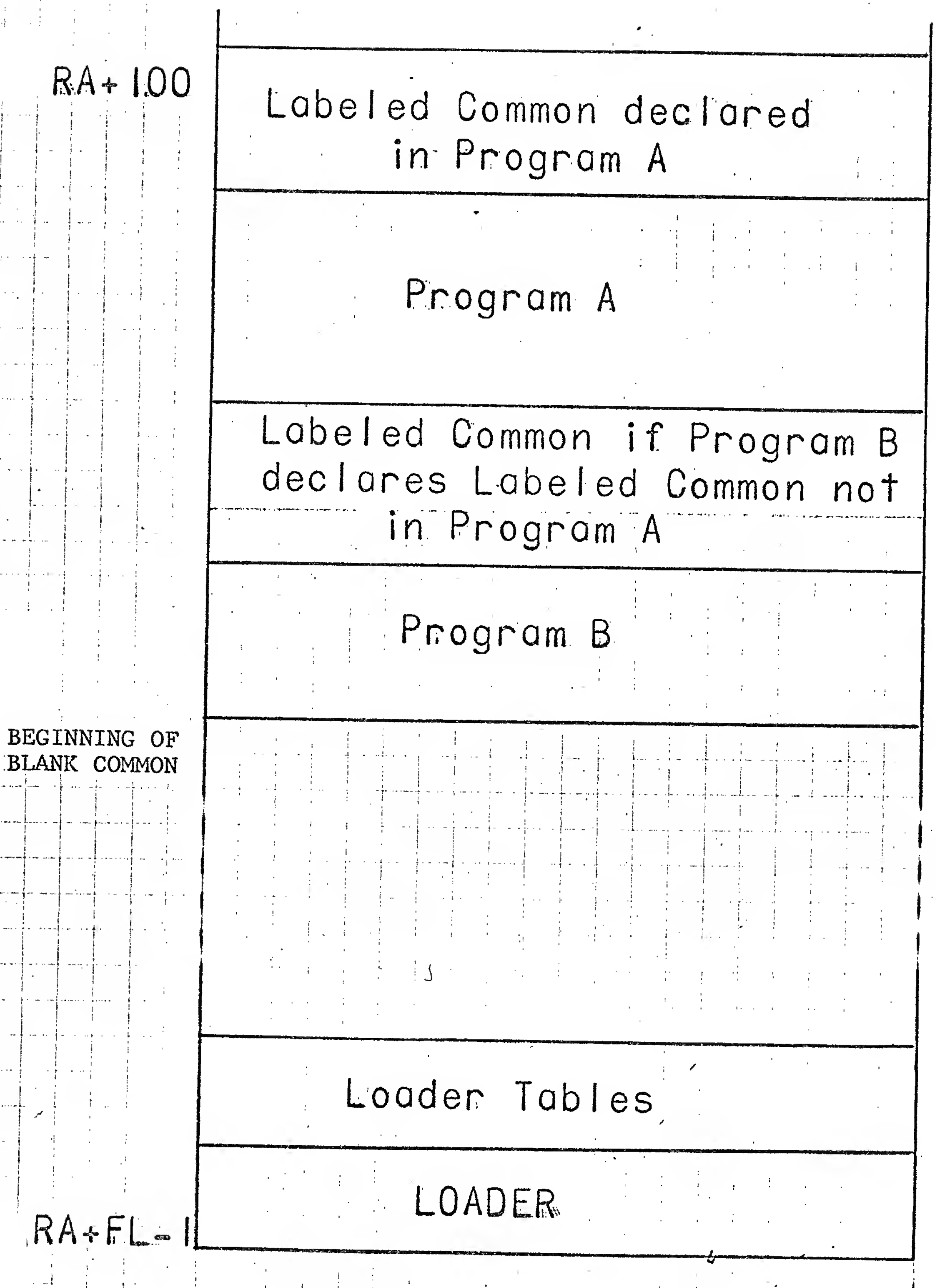
- A RSS BIT
- B NO MAP BIT
- C REQUEST EXIT FLAG
- D END OF LOAD
- E PARTIAL MAP BIT

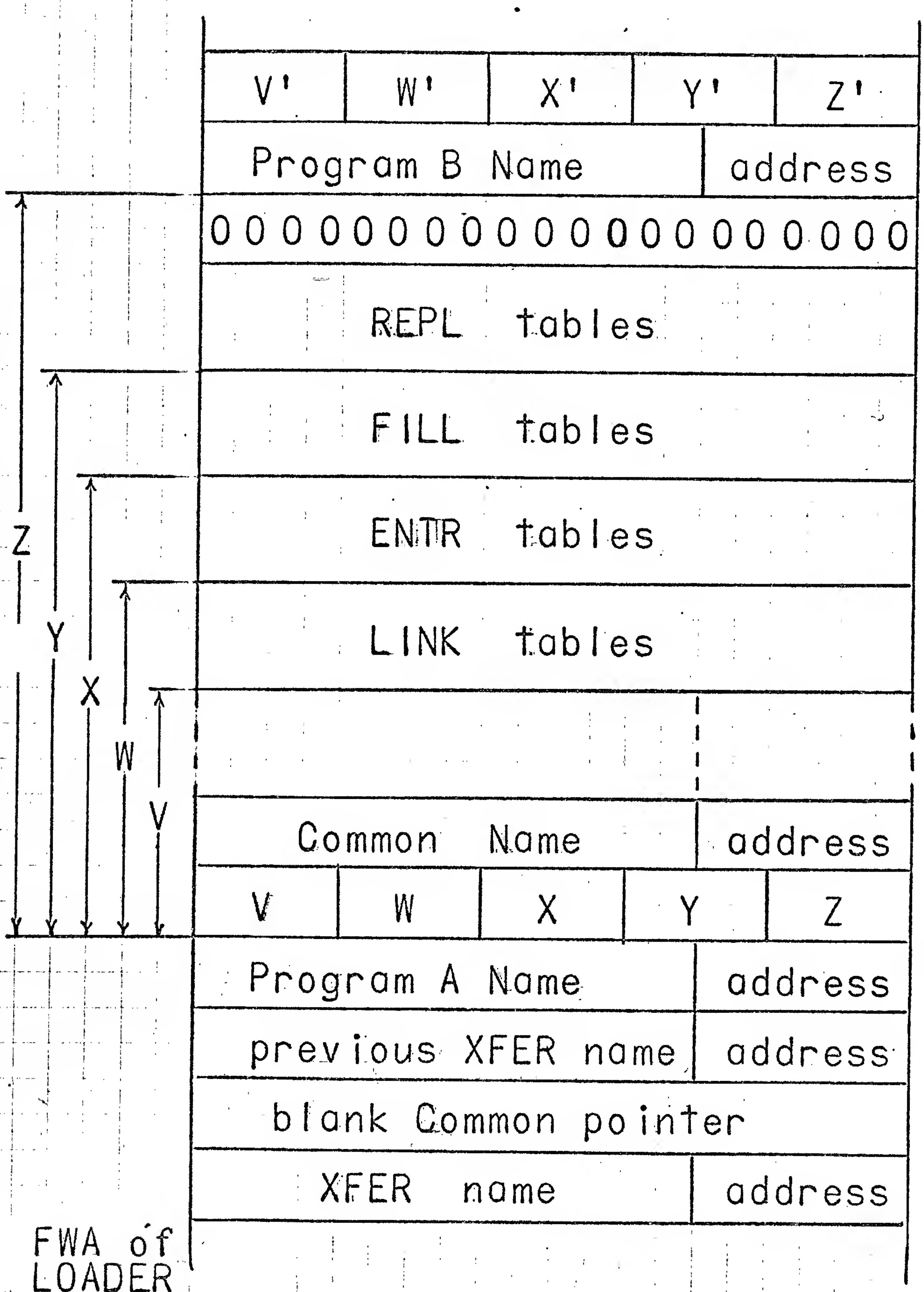
Y (3 BITS) LAST CONTROL CARD

- 0 PROGRAM CALL
- 1 LOAD
- 2 EXECUTE
- 4 NOGO

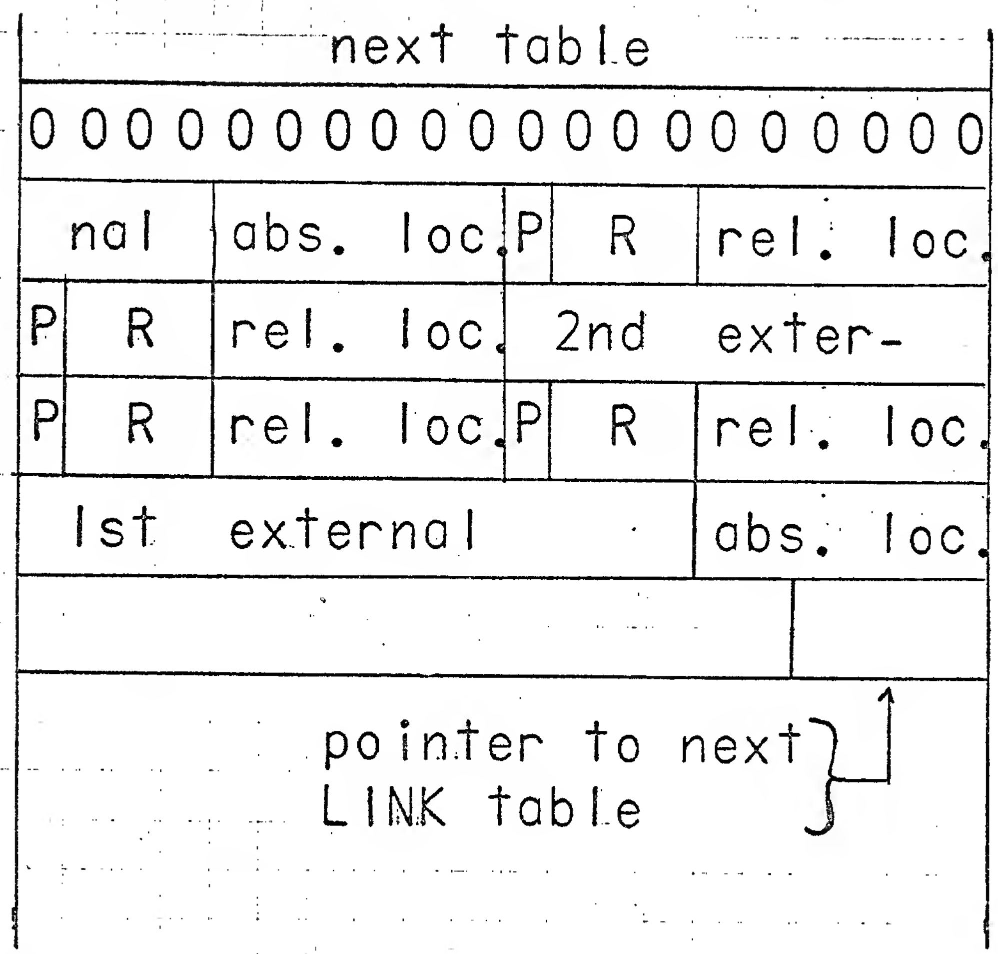
RA+100	* Section Name	000000	SECTION table
	Program Name	000000	
	⋮	⋮	
	Program Name	000000	
	00000000000000	000000	
	* Segzero Name	000000	SEGMENT table
	Program/Section Name	000000	
	⋮	⋮	
	Program/Section Name	000000	
	* Segment Name	000000	
	Program/Section Name	000000	
	⋮	⋮	
	Program/Section Name	000000	
	00000000000000	000000	
			FWA of Segzero

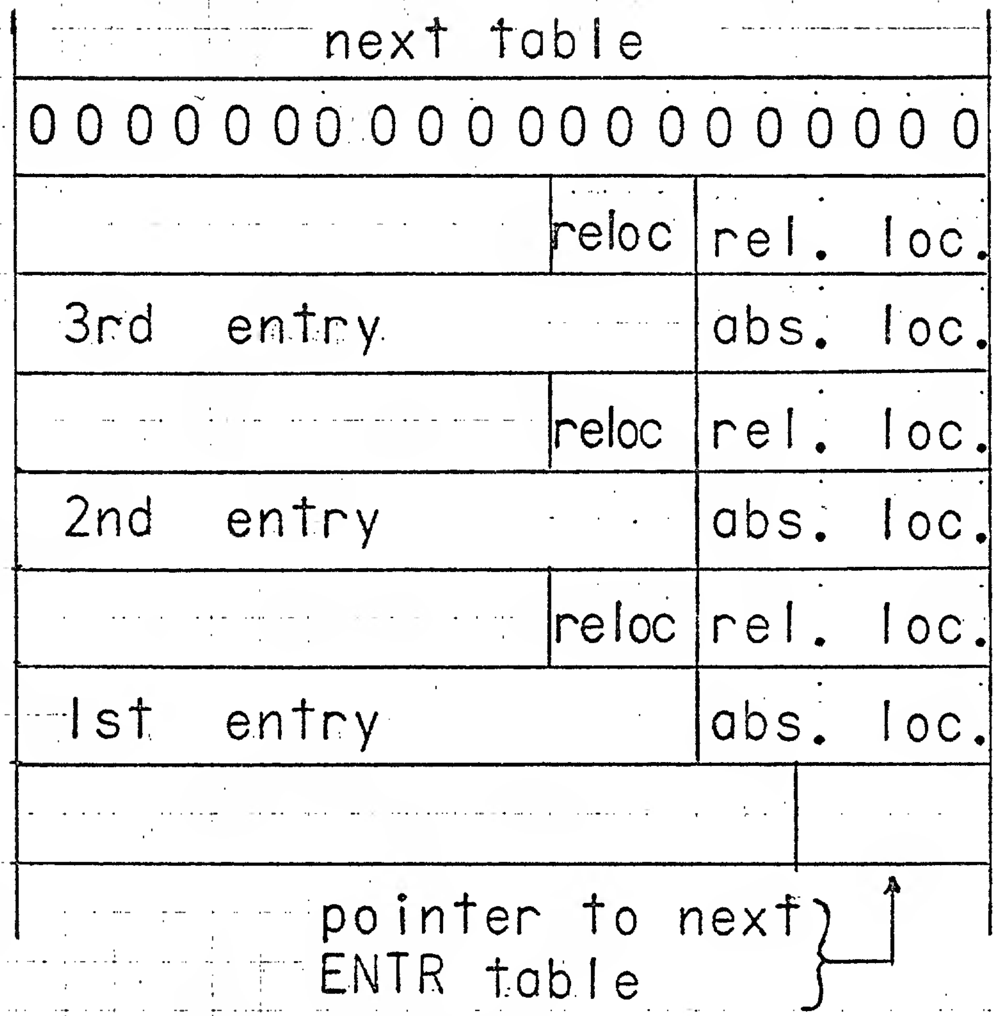
* $2^{59} = 1$


RA+100
Labeled Common declared in Program A
Program A
Labeled Common if Program B declares Labeled Common not in Program A
Program B
BEGINNING OF BLANK COMMON
Loader Tables
LOADER
RA+FL-1



V'
W'
X'
Y'
Z'
Program B Name
address
0000000000000000000
REPL tables
FILL tables
ENTR tables
LINK tables
Common Name
address
V
W
X
Y
Z
Program A Name
address
previous XFER name
address
blank Common pointer
XFER name
address
FWA of LOADER
Z
Y
X
W
V


next table
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
nal
abs. loc.
P
R
rel. loc.
P
R
rel. loc.
2nd exter-
P
R
rel. loc.
P
R
rel. loc.
1st external
abs. loc.
pointer to next
LINK table


next table
0000000000000000000
reloc
rel. loc.
3rd entry
abs. loc.
reloc
rel. loc.
2nd entry
abs. loc.
reloc
rel. loc.
1st entry
abs. loc.
pointer to next ENTR table

<table>
<tr><td colspan="6">next table</td></tr>
<tr><td colspan="6">0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td></tr>
<tr><td>P</td><td>R</td><td>rel. loc</td><td>P</td><td>R</td><td>rel. loc</td></tr>
<tr><td>P</td><td>R</td><td>rel. loc</td><td colspan="3">control byte</td></tr>
<tr><td colspan="3">control byte</td><td>P</td><td>R</td><td>rel. loc</td></tr>
<tr><td colspan="5"></td><td></td></tr>
</table>

pointer to next
FILL table

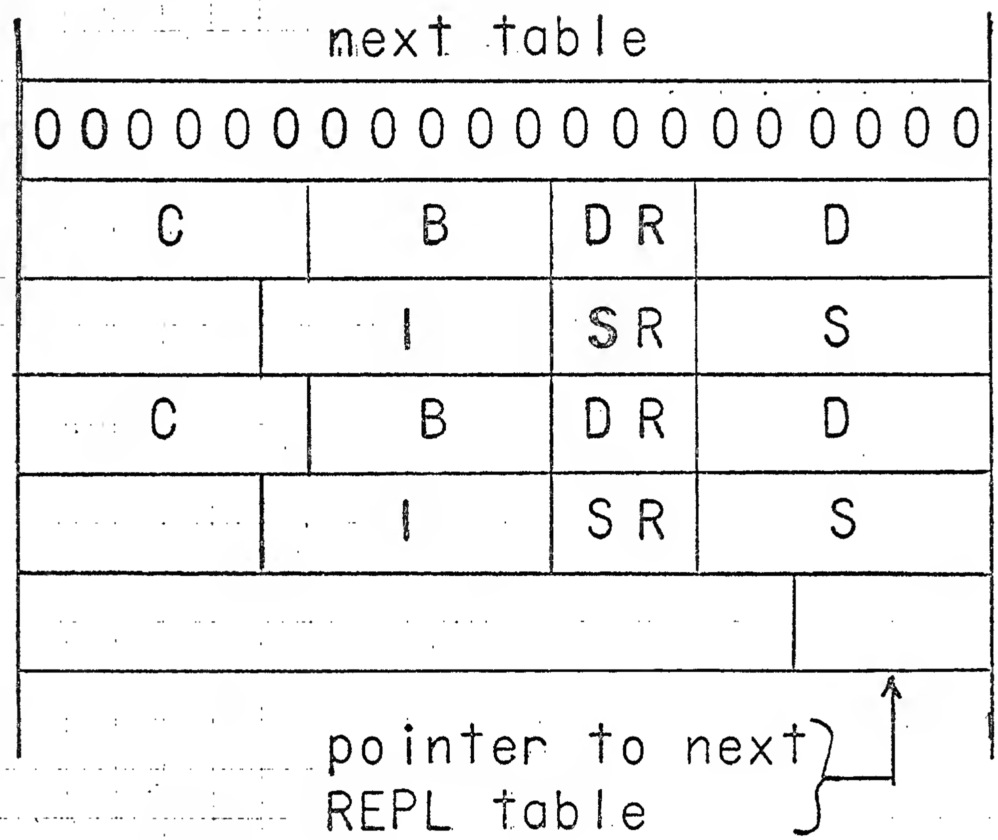

next table
0000000000000000000
C
B
DR
D
I
SR
S
C
B
DR
D
I
SR
S
pointer to next
REPL table

TABLE FORMAT

CN	00	WC	000	LR	L

CN   Code Number - identifies type of table

WC   Word Count - number of words in table
(total table length = WC+11)

LR and L used only with TEXT table

LABLE TABLE

7 7	0 0	0 0 1 6	0 0 0	0 0 0	0 0 0 0 0 0
Program Name					0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0					
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0					

PIDL TABLE

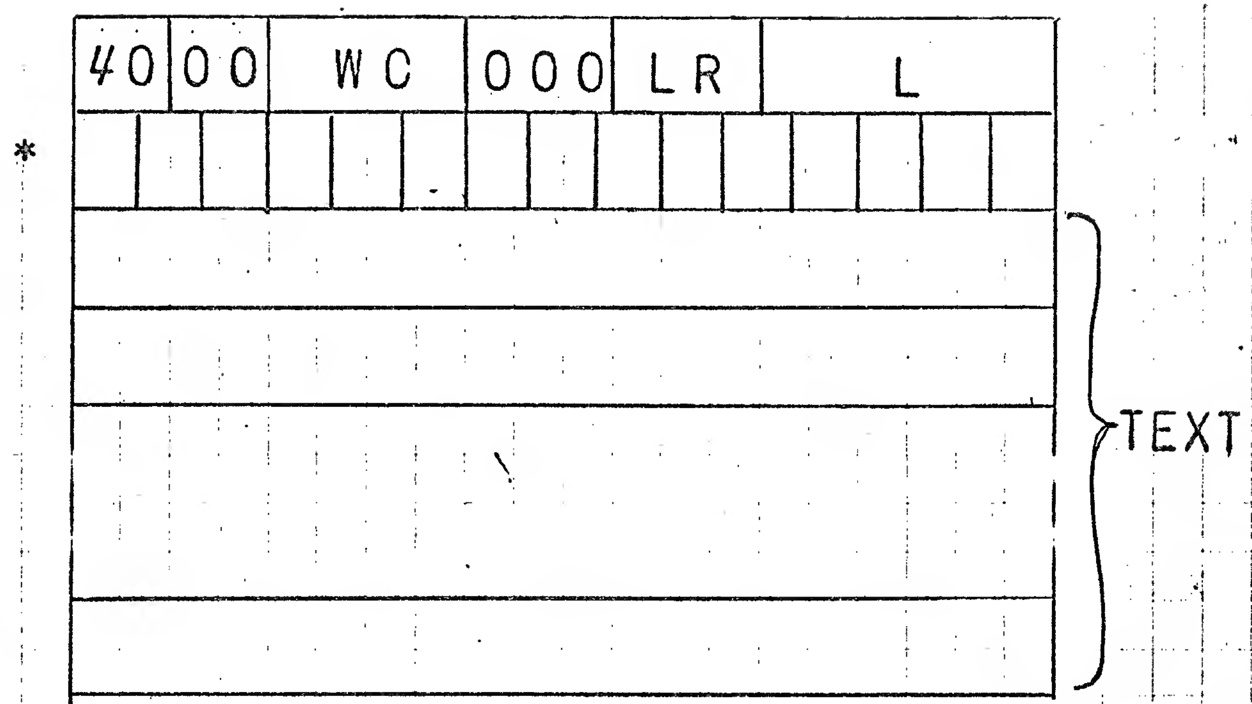
34	00	WC	000	000	000000
PROGRAM NAME					PROGRAM LENGTH
COMMON NAME					BLOCK LENGTH
COMMON NAME					BLOCK LENGTH
COMMON NAME					BLOCK LENGTH

First entry must be the program name

Blank Common same as any common (name is 55555555555555)

All names left justified with zero fill

TEXT TABLE



* Relocation bytes (4 bits)

WC   limited to maximum of 16
      more text requires more
      TEXT tables

LR   specifies area into which text is
      to be loaded
        0 = absolute (relative to RA)
        1 = program
        3 - n = common

L   first word address for text
      within LR area

RELOCATION BYTES

1000	upper+
1100	upper-
1010	upper+ lower+
1011	upper+ lower-
1110	upper- lower+
1111	upper- lower-
0010	lower+
0011	lower-
0100	middle+
0110	middle-
0000	no relocation

FILL TABLE

<table>
<tr><td colspan="2">42</td><td>00</td><td colspan="2">WC</td><td colspan="2">000</td><td>000</td><td colspan="2">000000</td></tr>
<tr><td colspan="4">0000000</td><td colspan="2">reloc</td><td>P</td><td>R</td><td colspan="2">L</td></tr>
<tr><td>P</td><td colspan="2">R</td><td colspan="3">L</td><td>P</td><td>R</td><td colspan="2">L</td></tr>
<tr><td>P</td><td colspan="2">R</td><td colspan="3">L</td><td colspan="3">0000000</td><td>reloc</td></tr>
<tr><td colspan="10"></td></tr>
<tr><td colspan="6"></td><td>P</td><td>R</td><td colspan="2">L</td></tr>
</table>

reloc - relocation
- 0 absolute
- 1 program
- 2 negative prog.
- 3-n common

R - area
- 0 absolute
- 1 program
- 3-n common

L - first word address in area

P - position in address
- 100 lower
- 101 middle
- 110 upper

```
        IDENT   PROG
        ORG     2
P1      VFD     30/5LINPUT,12/0,18/BE1
P2      VFD     36/6LOUTPUT,6/0,18/BE2
P3      VFD     42/7LSCRATCH,18/BE3
        USE     /ALPHA/
AL1     BSS     2000B
AL2     BSS     2000B
        USE     /BETA/
BE1     BSS     5
BE2     BSS     5
BE3     BSS     5
        USE     //
BUFFER  BSS     2000B
        USE
        SA1     P1-2
        SB1     A1
        MX3     42
LOOP    SA1     A1+B1
        ZR      X1,OUT
        BX6     X1*X3
        SA6     X1
```

```
        ZR      LOOP
        USE     /BETA/
        ORG     0
        VFD     30/5LINPUT,30/0
        VFD     60/ALI
        VFD     60/AL2
        VFD     36/6LOUTPUT, 24/0
        VFD     60/AL2
        VFD     60/AL2+2000B
        VFD     42/7LSCRATCH,18/0
        VFD     60/BUFFER
        VFD     60/BUFFER+2000B
        USE     *
OUT
```

	3	4	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	2	2	1	7	0	7	0	0	0	0	0	0	length					
	0	1	1	4	2	0	1	0	0	1	0	0	0	0	0	0	4	0	0	0
	0	2	0	5	2	4	0	1	0	0	0	0	0	0	0	0	0	0	1	7
	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	0	2	0	0	0
	4	0	0	0	0	0	0	7	0	0	0	0	0	1	0	0	0	0	0	2
*	0	0	0	0	4	1	0	4	0	0	0	0	0	0	0	0	0	0	0	0
	1	1	1	6	2	0	2	5	2	4	0	0	0	0	0	0	0	0	0	0
	1	7	2	5	2	4	2	0	2	5	2	4	0	0	0	0	0	0	0	5
	2	3	0	3	2	2	0	1	2	3	0	3	1	0	0	0	0	0	1	2
	5	1	1	0	0	0	0	0	0	0	6	4	1	1	0	4	3	3	5	2
	5	4	1	1	1	0	3	0	1	0	0	0	0	1	0	1	1	6	1	3
	5	3	6	1	0	0	4	0	0	0	0	0	0	0	5	4	6	0	0	0
	4	0	0	0	0	0	1	2	0	0	0	0	0	4	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	1	1	6	2	0	2	5	2	4	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
	1	7	2	5	2	4	3	0	2	5	2	4	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0

* 0 | 0 | 0 | 0 | 4 | 1 | 0 | 4 | 0 | 0

000000000000100001000100000000

no | no | no | up | mid | mid | no

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0
2	3	0	3	2	2	0	1	2	4	0	3	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
4	2	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	4	0	0	4	0	0	0	0	0	1
4	0	0	4	0	0	0	0	0	2	4	0	0	4	0	0	0	0	0	4
4	0	0	4	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	4
4	0	0	1	0	0	0	0	0	2	4	0	0	1	0	0	0	0	0	3
4	0	0	1	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	5
4	0	0	4	0	0	0	0	0	7	4	0	0	4	0	0	0	0	1	0
4	0	0	0	x	x	x	x	0	0	0	0	0	1	0	0	0	0	1	0

LINK TABLE

<table>
<tr><td colspan="2">44</td><td>00</td><td>WC</td><td colspan="2">000</td><td>000</td><td>000000</td></tr>
<tr><td colspan="7">name of external</td><td>000000</td></tr>
<tr><td>P</td><td colspan="2">R</td><td colspan="2">L</td><td>P</td><td>R</td><td>L</td></tr>
<tr><td>P</td><td colspan="2">R</td><td colspan="2">L</td><td colspan="3">name of ext-</td></tr>
<tr><td colspan="3">ernal</td><td colspan="2">000000</td><td>P</td><td>R</td><td>L</td></tr>
<tr><td colspan="8"></td></tr>
<tr><td>P</td><td colspan="2">R</td><td colspan="2">L</td><td>P</td><td>R</td><td>L</td></tr>
</table>

P position in word
- 4 lower
- 5 middle
- 6 upper

R area
- 0 absolute
- 1 program
- 3-n common

L location reference made from

ENTR TABLE

<table>
<tr><td>36</td><td>00</td><td>WC</td><td>000</td><td>000</td><td>000000</td></tr>
<tr><td colspan="5">name of entry point</td><td>000000</td></tr>
<tr><td colspan="4">000000000000</td><td>R</td><td>L</td></tr>
<tr><td colspan="5">name of entry point</td><td>000000</td></tr>
<tr><td colspan="4">000000000000</td><td>R</td><td>L</td></tr>
<tr><td colspan="6"></td></tr>
<tr><td colspan="5">name of entry point</td><td>000000</td></tr>
<tr><td colspan="4">000000000000</td><td>R</td><td>L</td></tr>
</table>

R area
- 0 absolute
- 1 program
- 3-n common

L location in area

```
        IDENT   GREEK
        ENTRY   ALPHA,BETA,GAMMA
        EXT     ALEPH,BETH,GIMMEL
ALPHA   DATA    0
        RJ      ALEPH
DELTA   SX6     B7
        SA6     A0
        RJ      GIMMEL
        SA1     A0
        JP      ALPHA
BETA    DATA    0
SA1     BETA
        BX6     X1
        SA6     ALPHA
        RJ      BETH
        JP      DELTA
GAMMA   DATA    0
        SA1     GAMMA
        BX6     X1
        SA6     GIMMEL
        JP      GIMMEL+1
```

3 4 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 7 2 2 0 5 0 5 1 3 0 0 0 0 length

3 6 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 4 2 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 2 0 5 2 4 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 4
0 7 0 1 1 5 1 5 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0

4 4 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 4 0 5 2 0 1 0 0 0 0 0 0 0 0 0 0 0
6 0 0 1 0 0 0 0 0 1 0 2 0 5 2 4 1 0 0 0
0 0 0 0 0 0 0 0 0 0 4 0 0 1 0 0 0 0 0 6
0 7 1 1 1 5 1 5 0 5 1 4 0 0 0 0 0 0 0 0
4 0 0 1 0 0 0 0 0 2 6 0 0 1 0 0 0 0 1 2
4 0 0 1 0 0 0 0 1 2 0 0 0 0 0 0 0 0 0 0

4 0 0 0 x x x x 0 0 0 0 0 1 0 0 0 0 0 0
* 0 0 0 0 2 0 1 0 4 2 0 0 4 0 x x x x x x
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 4 6 0 0 0 4 6 0 0 0
7 6 6 7 0 5 4 6 0 0 0 1 0 0 0 0 0 0 0 0
5 4 1 0 0 0 2 0 0 0 0 0 0 0 0 4 6 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

* 0 | 0 | 0 | 0 | 2 | 0 | 1 | 0 | 4 | 2 | 0 | 0 | 4 | 0
000000000000000 1000000 1000 1000 10000000 100000
no | no | no | mid | no | up | up | up | no | up | no

5	1	1	0	0	0	0	0	0	4	1	0	6	1	0	4	6	0	0	0
5	1	6	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0	2	4	6	0	0	0	4	6	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	1	0	0	0	0	0	1	0	1	0	6	1	0	4	6	0	0	0
5	1	6	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	1

REPL TABLE

43	00	WC	000	000	000000
		I		SR	S
C		B		DR	D
		I		SR	S
C		B		DR	D

SR source area
S address in area
B block size
DR destination area
D address in area
C number of replications
I replication interval
SR and DR
- 0 absolute
- 1 program
- 3-n common

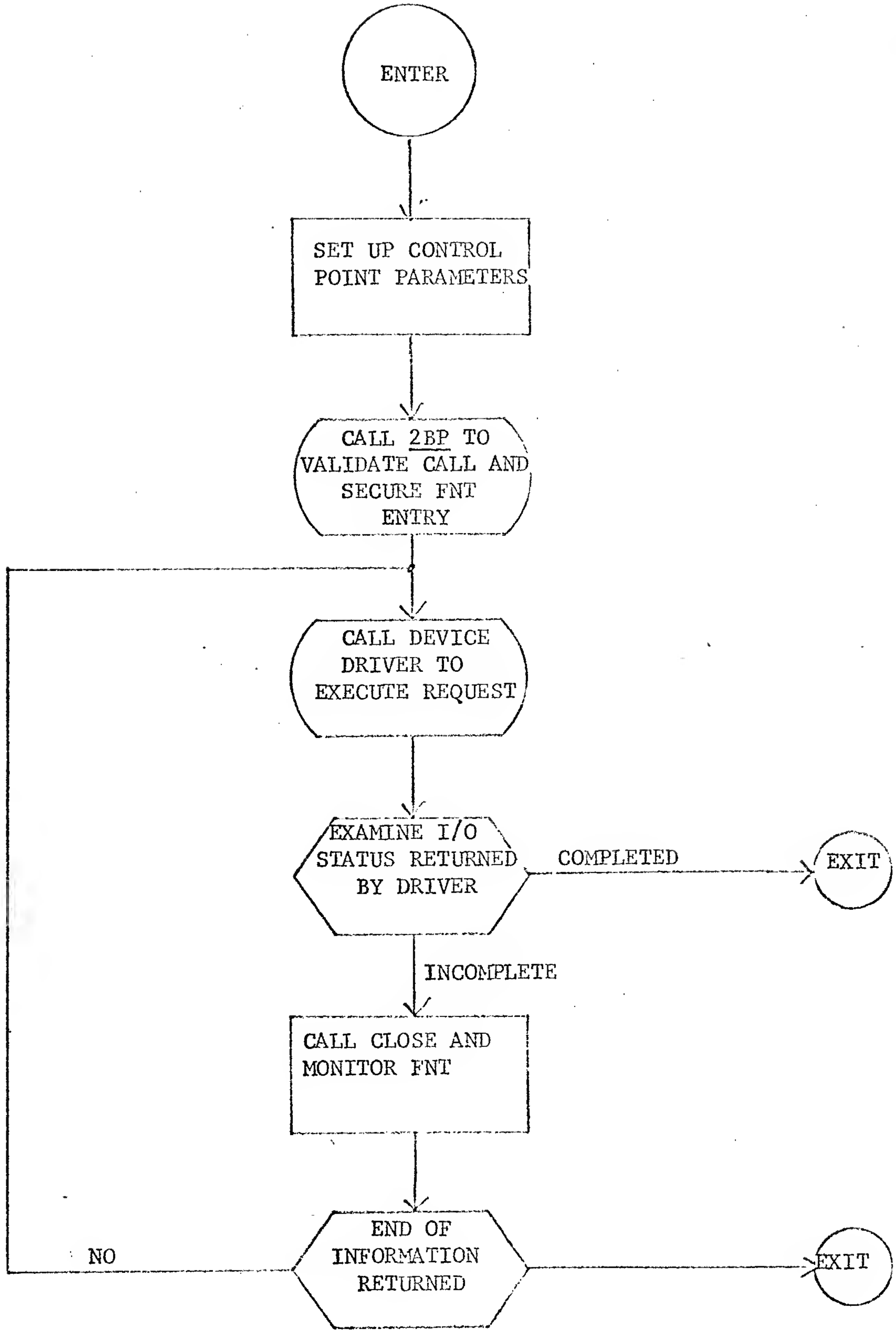
XFER TABLE

4 6	0 0	0 0 0 1	M M	D D	Y Y
name				0 0 0 0 0 0	

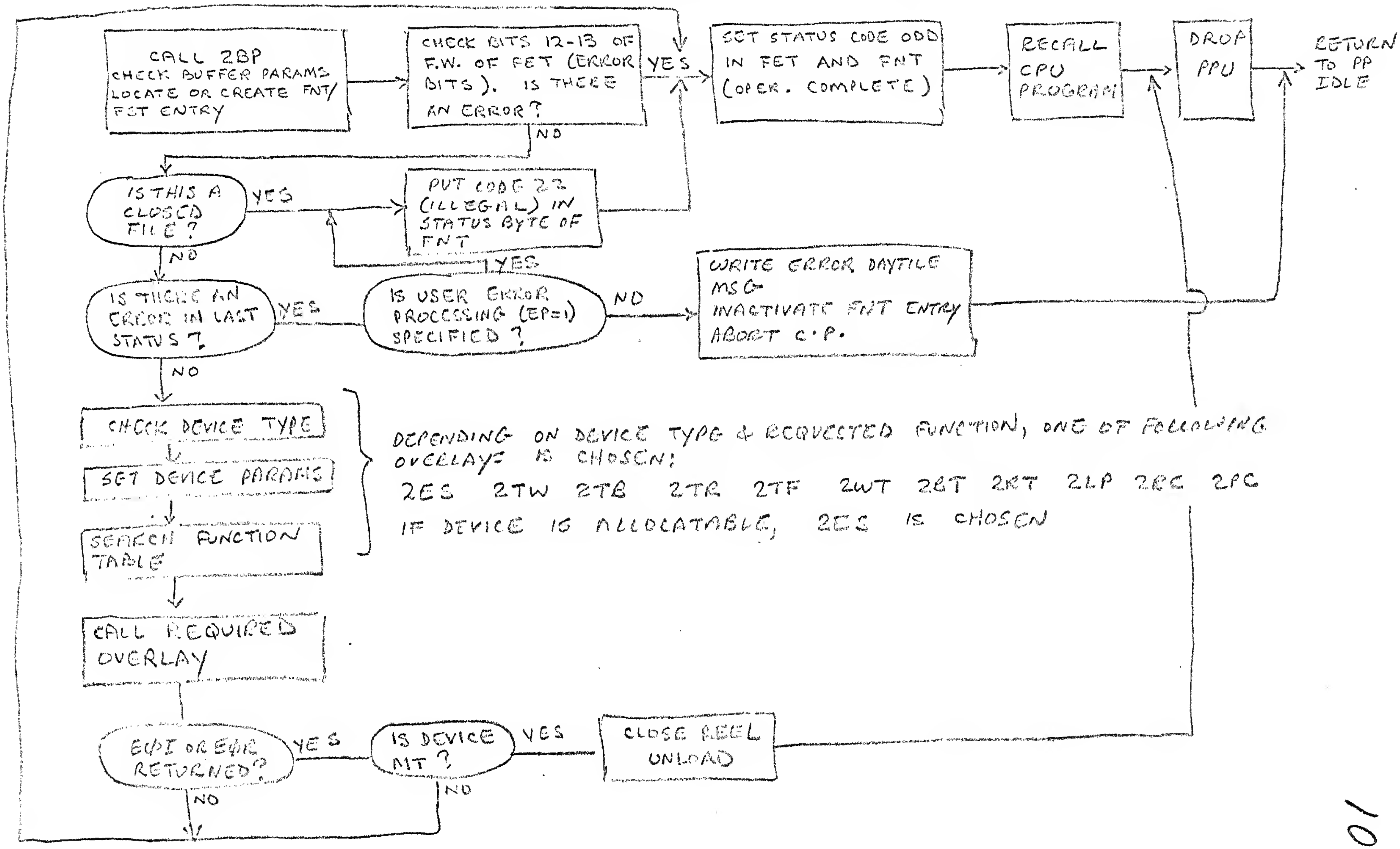
ABSOLUTE OVERLAY

50	00	L1	L2	LOAD ADDRESS	ENTRY ADDRESS

words 2 through end of logical record are absolute text

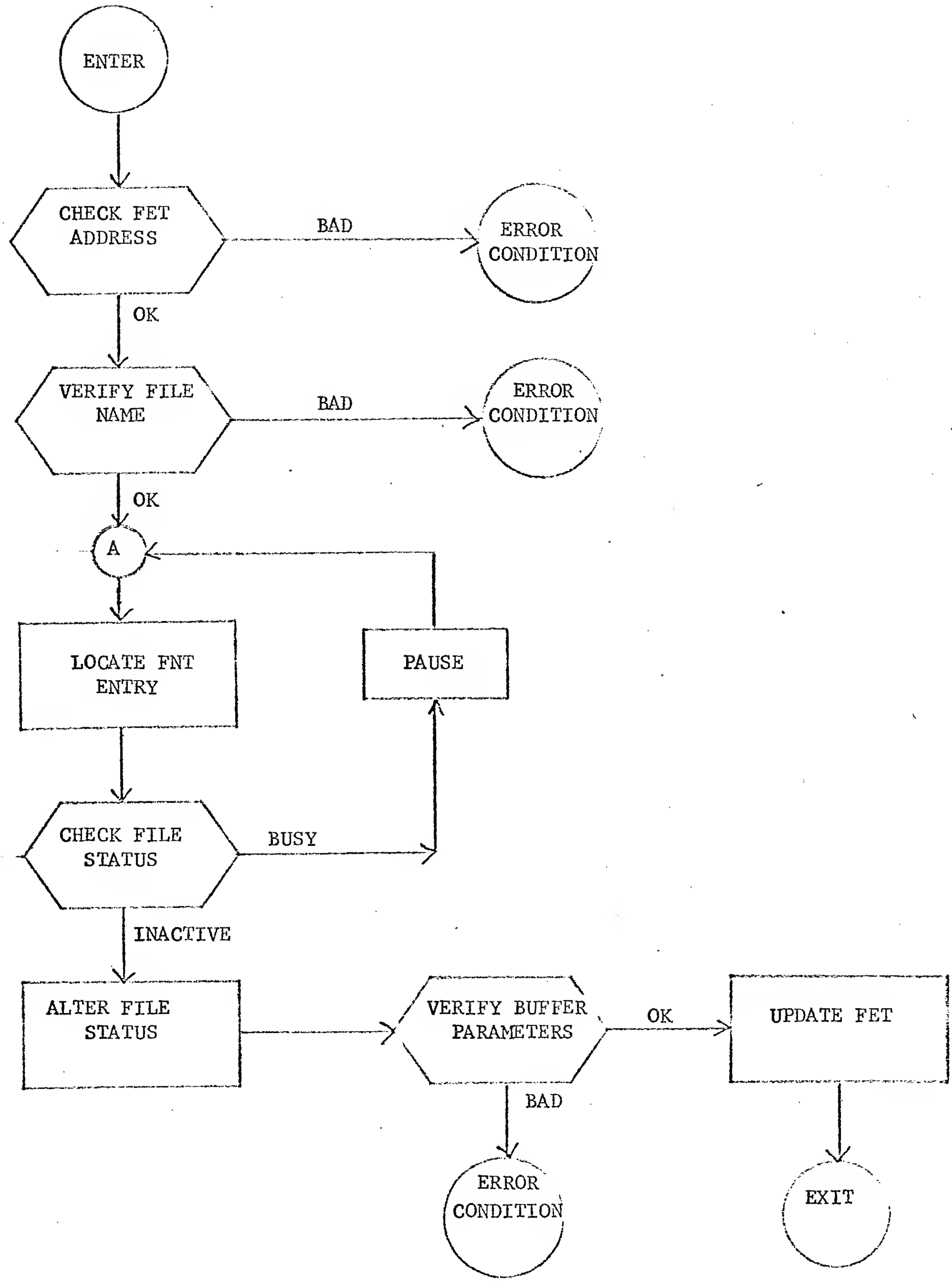

ENTER
SET UP CONTROL POINT PARAMETERS
CALL 2BP TO VALIDATE CALL AND SECURE FNT ENTRY
CALL DEVICE DRIVER TO EXECUTE REQUEST
EXAMINE I/O STATUS RETURNED BY DRIVER
COMPLETED
EXIT
INCOMPLETE
CALL CLOSE AND MONITOR FNT
END OF INFORMATION RETURNED
NO
EXIT

CIØ (CIRCULAR I/Ø)



10-1A

2BP

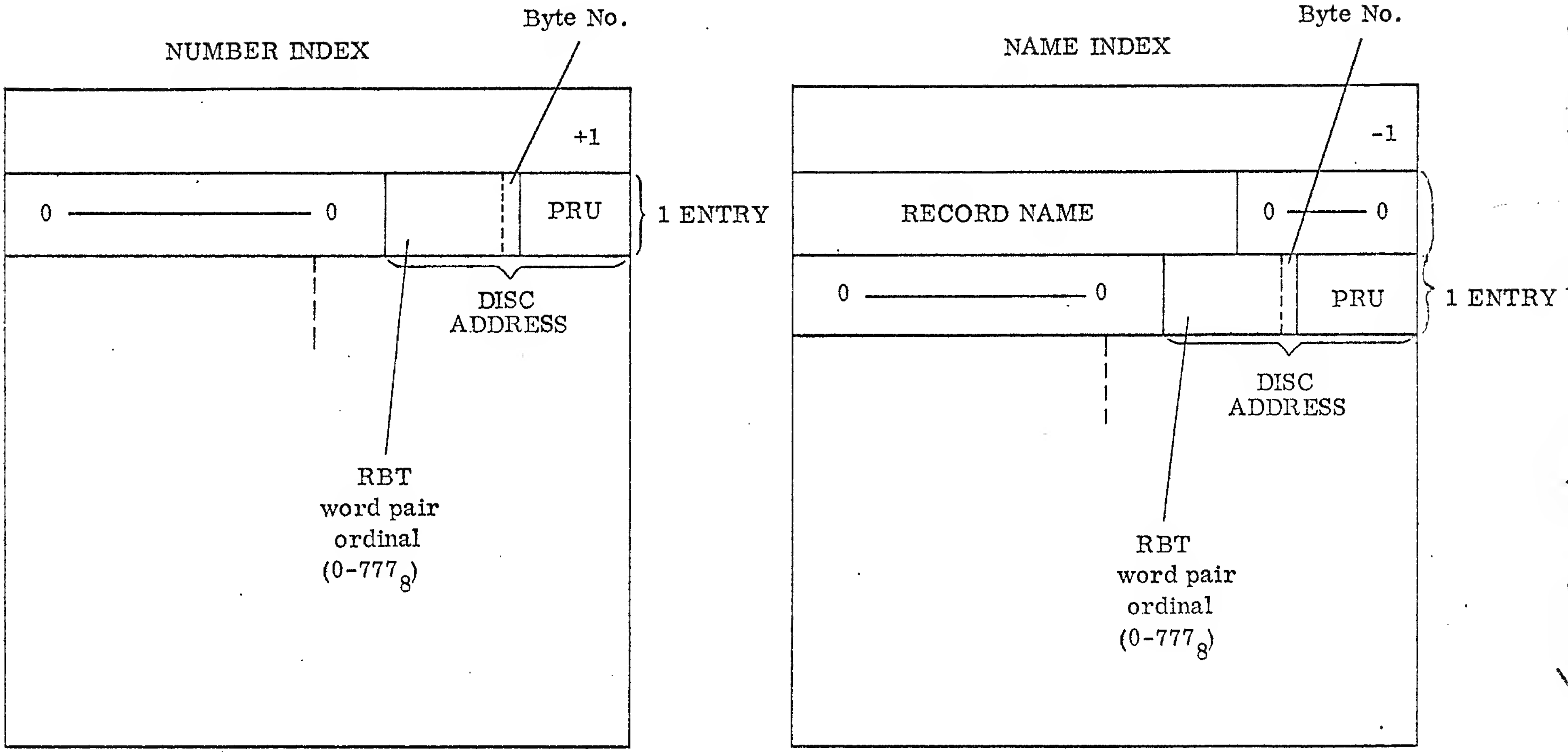

ENTER
CHECK FET ADDRESS
BAD
ERROR CONDITION
OK
VERIFY FILE NAME
BAD
ERROR CONDITION
OK
A
LOCATE FNT ENTRY
PAUSE
CHECK FILE STATUS
BUSY
INACTIVE
ALTER FILE STATUS
VERIFY BUFFER PARAMETERS
OK
UPDATE FET
BAD
ERROR CONDITION
EXIT

FILE ENVIRONMENT TABLE

Bits 59 47 44 35 32 29 23 17 0

Words									
1	logical file name (lfn)	code and status							
2	device type	r	n	u p	e p		disposition code	ℓ	FIRST
3	0	IN							
4	0	OUT							
5	FNT pointer	record block size	physical record unit size	LIMIT					
6		working storage fwa		working storage lwa+1					
7		record request/return information							
8	record number		index length	index address					
9		EOI address		error address					
10	Label file name (first 10 chars)								
11	Label file name (last 10 chars)								
12	edition number	retention cycle	creation date						
13	position number	multi-file name	reel number						

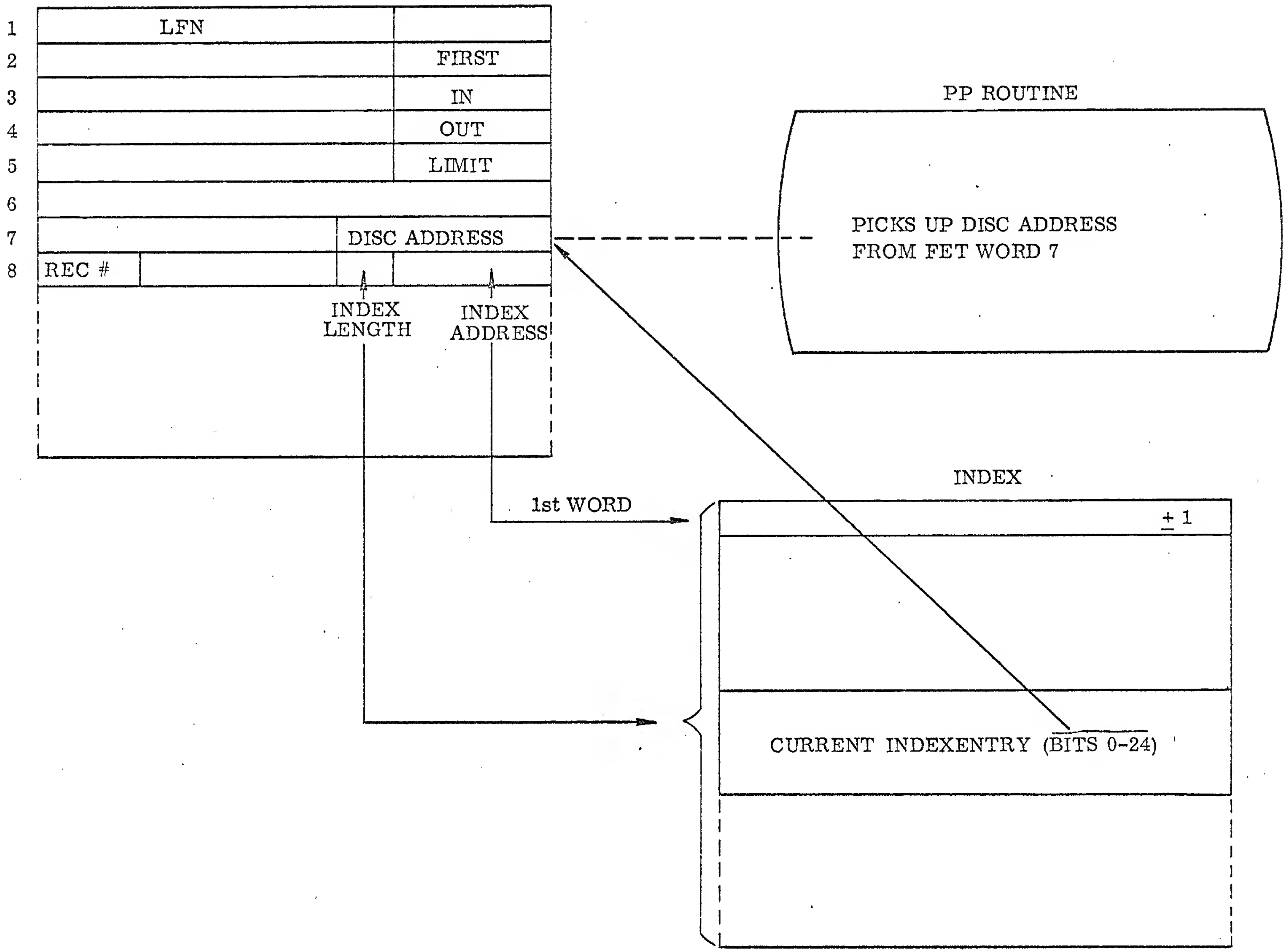
RANDOM ACCESS FILE INDEXES



SCOPE I/O (TAPE + DISC)

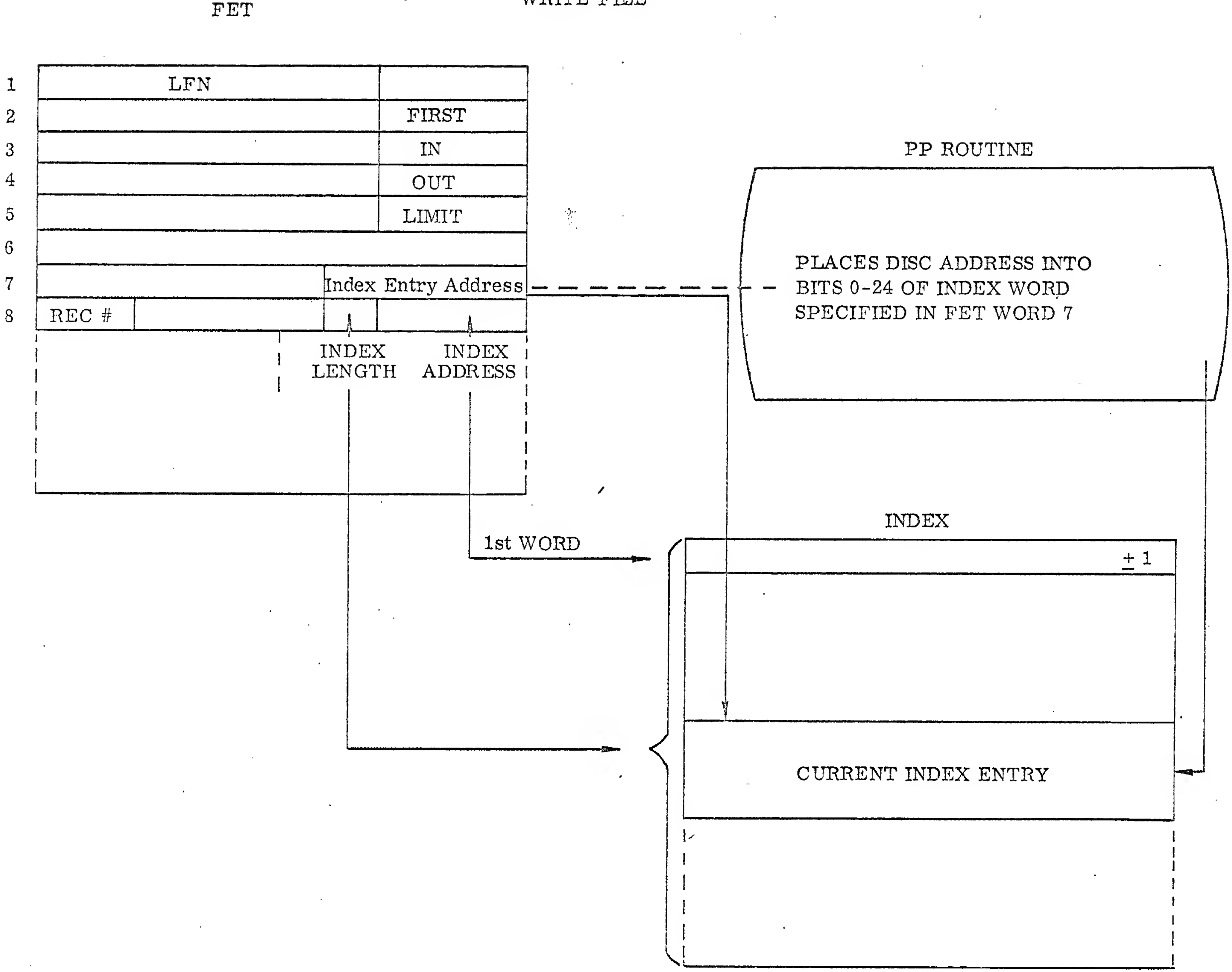
10-4

READ FILE

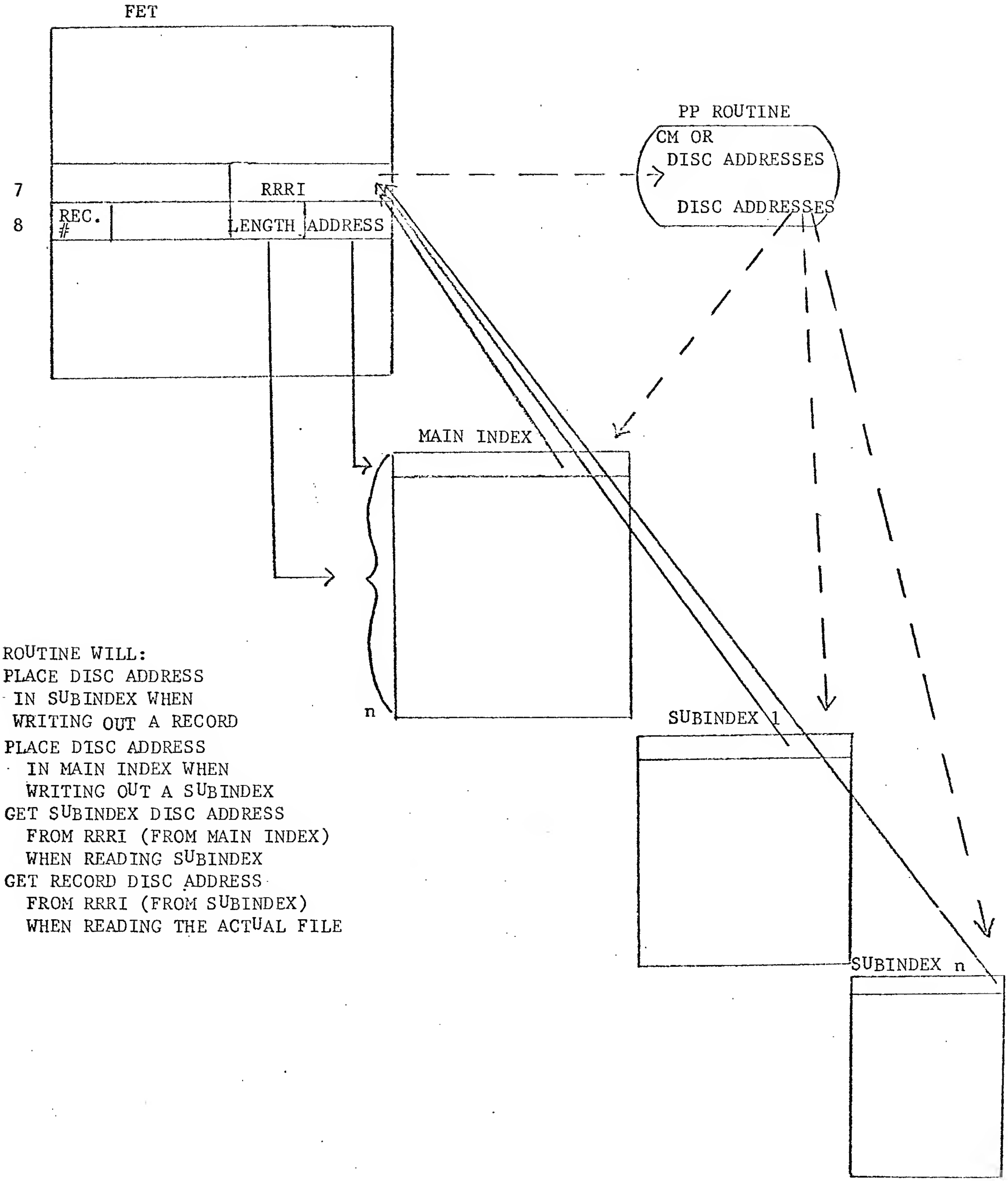


10-5

WRITE FILE


FET
1
2
3
4
5
6
7
8
LFN
FIRST
IN
OUT
LIMIT
Index Entry Address
REC #
INDEX LENGTH
INDEX ADDRESS
PP ROUTINE
PLACES DISC ADDRESS INTO
BITS 0-24 OF INDEX WORD
SPECIFIED IN FET WORD 7
1st WORD
INDEX
± 1
CURRENT INDEX ENTRY

MULTI INDEX



PP ROUTINE WILL:
- PLACE DISC ADDRESS IN SUBINDEX WHEN WRITING OUT A RECORD
- PLACE DISC ADDRESS IN MAIN INDEX WHEN WRITING OUT A SUBINDEX
- GET SUBINDEX DISC ADDRESS FROM RRRI (FROM MAIN INDEX) WHEN READING SUBINDEX
- GET RECORD DISC ADDRESS FROM RRRI (FROM SUBINDEX) WHEN READING THE ACTUAL FILE

MAJOR TAPE ROUTINES

CIO: GENERAL CHECKING AND ROUTING ROUTINE FOR ANY INPUT/OUTPUT REQUEST.

2TR: READ DRIVER FOR ½" MAGNETIC TAPE TRANSPORTS.

2TW: WRITE DRIVER FOR ½" MAGNETIC TAPE TRANSPORTS.

2TB: POSITION BACKWARD ROUTINE FOR ½" MAGNETIC TAPE.

2TF: POSITION FORWARD ROUTINE FOR '" MAGNETIC TAPE.

4LB: READ AND WRITE ROUTINE FOR ½" MAGNETIC TAPE WHICH PROCESSES LABEL INFORMATION.

OPEN: GENERAL FILE INITIALIZING ROUTINES. FOR LABELED TAPES, OPEN CALLS THE APPROPRIATE ROUTINES TO POSITION TAPE AND READ OR WRITE LABELS.

CLOSE: GENERAL FILE TERMINATING ROUTINE. FOR LABELED TAPES, CLOSE CALLS THE APPROPRIATE ROUTINES TO POSITION TAPE AND READ OR WRITE LABELS.

NOTES:

DIFFERENCE IN PHYSICAL AND LOGICAL TAPE MARKS

	HOW WRITTEN	HOW RECORDED	WHEN WRITTEN	ACTION ON READING
PHYSICAL EOF (TAPE MARK)	DEVICE FUNCTION	1 CHAR. 17_8 EVEN PARITY and 6" gap	IN CONJUNCTION WITH TAPE LABEL [before & after]	EOI & EOF RETURNED TO CIO
LOGICAL EOF	DATA WRITE	8 CHAR. BIN / BCD 0000 / 2020 0000 / 2020 0017 / 2017	WRITEF	SETS EOF BITS IN CODE & STATUS

SINGLE REEL FILE

VOL1
HDR1
TAPE MARK
DATA
TAPE MARK
EOF1
TAPE MARK
TAPE MARK

MULTI-REEL FILE

VOL1
HDR1
TAPE MARK
FIRST VOLUME DATA
TAPE MARK
EOV1
TAPE MARK
TAPE MARK

•
•
•

VOL1
HDR1
TAPE MARK
LAST VOLUME DATA
TAPE MARK
EOF1
TAPE MARK
TAPE MARK

MULTI-FILE REEL

VOL1
HDR1
TAPE MARK
FILE A DATA
TAPE MARK
EOF1
TAPE MARK
HDR1
TAPE MARK
FILE B DATA
TAPE MARK
• • •
TAPE MARK
EOF1
TAPE MARK
TAPE MARK

NOTES:

10-12

MULTI-REEL MULTI-FILE

REEL 1

- VOL 1
- HDR 1
- TAPE MARK
- FILE A
- TAPE MARK
- EOF1
- TAPE MARK
- HDR1
- TAPE MARK
- FILE B
- TAPE MARK
- EOV1
- TAPE MARK
- TAPE MARK

REEL 2

- VOL 1
- HDR 1
- TAPE MARK
- FILE B CONTINUED
- TAPE MARK
- EOV1
- TAPE MARK
- TAPE MARK

REEL 3

- VOL 1
- HDR 1
- TAPE MARK
- LAST OF FILE B
- TAPE MARK
- EOF1
- TAPE MARK
- HDR1
- TAPE MARK
- FILE C
- TAPE MARK
- EOV1
- TAPE MARK
- TAPE MARK

10-13

LABELS - 80 CHARACTERS

VOLUME HEADER LABEL

3 CHAR.	1 CHAR.	6 CHAR.	1 CHAR.	1 CHAR.	60 CHAR.
V O L	1	VISUAL REEL NUMBER (VRN)	BLANK / NON BLANK	BLANK 0 1 2	19 SPACES FOR OPERATING SYSTEM; 49 SPACES FOR FUTURE STAN.
LABEL IDENT.	LABEL NUMBER		SECURITY	DENSITY	

FILE HEADER LABEL

1 3 CHAR.	1 CHAR.	20 CHAR.
H D R	1	FILE LABEL NAME
LABEL IDENTIFIER	LABEL	

21	25 3 CHAR.	28 4 CHAR.	32 4 CHAR.	36 3 CHAR.	
	MULTI FILE IDENTIFICATION (MFN)	REEL NUMBER	MULTI-FILE POSITION NUMBER	RESERVED "SPACES"	ED. NO.

41	42	43 5 CHAR.	48	49 5 CHAR.	54	55 6 CHARS.
		CREATION DATE YYDDD		EXPIRATION DATE YYDDD		BLOCK COUNT "ZEROS"
	"SPACE"		"SPACE"		SECURITY	

61 20 CHARACTERS
RESERVED FOR FUTURE STANDARDIZATION "SPACES"

10-14

FILE AND VOLUME TRAILER LABELS

1 3 CHAR	1 CHAR.	20 CHAR. 20
EOF or EOV	1	FILE LABEL NAME
LABEL IDENTIFIER	LABEL #	

21	25 3 CHAR.	28 4CHAR.	36 3CHAR.		40
→	MULTI FILE IDENTIFICA-TION (MFN)	REEL NUMBER	MULTI FILE POSITION NUMBER	RESERVED "SPACES"	ED. NO.

41	42	43 5 CHAR.	48	49 5 CHAR.	54	55 6 CHAR. 60
→	SPACE	CREATION DATE YYDDD	SPACE	EXPIRATION DATE YYDDD	SECURITY	BLOCK COUNT NON ZERO = PRU COUNT

61 20 CHARACTERS 80
← RESERVED FOR FUTURE STANDARDIZATION → "SPACES"

10-15

FET TAPE LABEL INFORMATION

Word	59	47	41	29	23	17 – 0
1	LOGICAL FILE NAME					
2		UP EP				FIRST
3						IN
4						OUT
5						LIMIT
6						
7						
8						
9						
10	FILE LABEL NAME (FIRST 10 CHARACTERS)					
11	FILE LABEL NAME (LAST 10 CHARACTERS)					
12	EDITION NUMBER	RETENTION CYCLE		CREATION DATE		
13	POSITION NUMBER		MULTI-FILE NAME		REEL NUMBER	

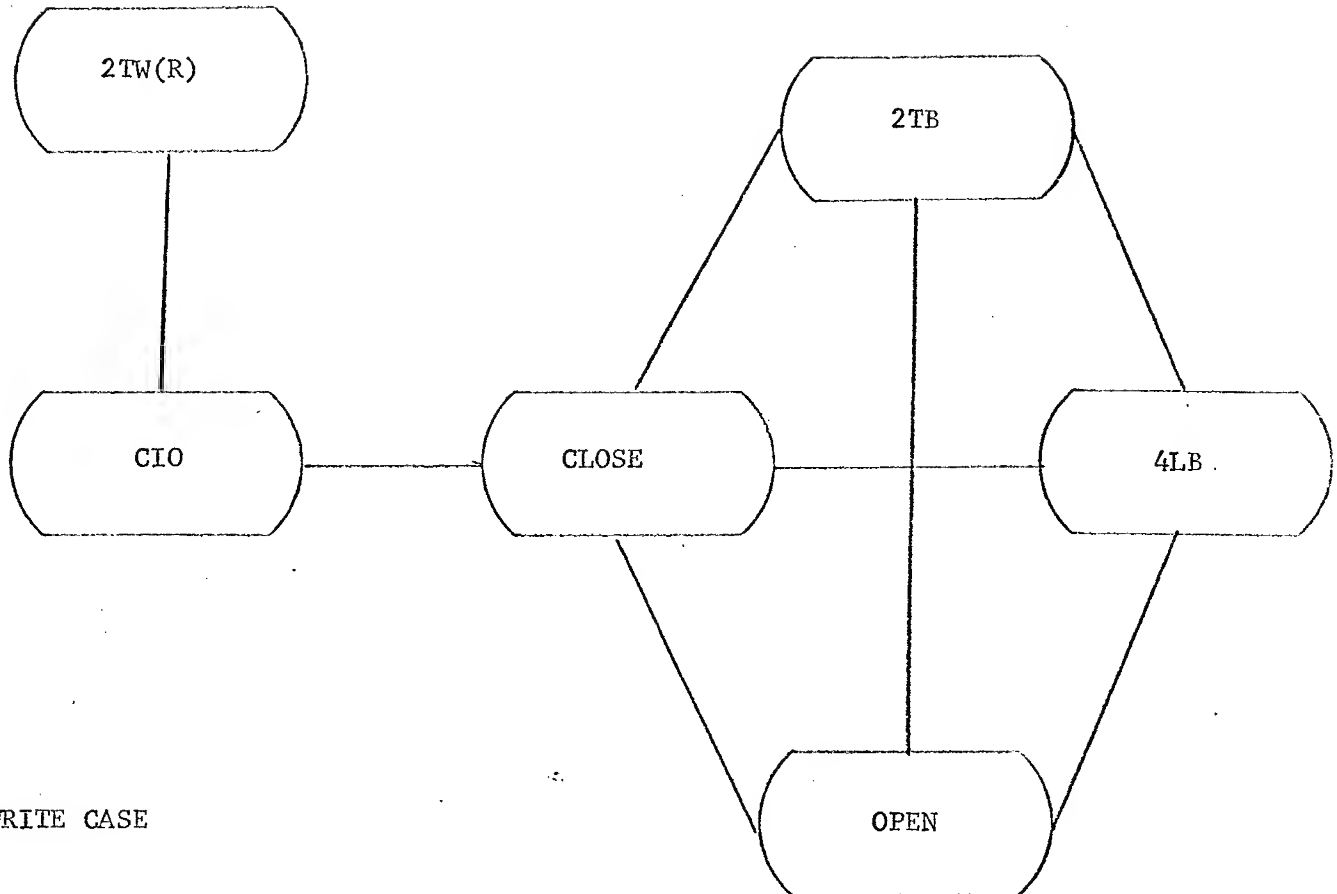
7-23 → 7-25

LABEL REQUESTS

REQUEST, LFILE, N, 2LO

REQUEST, MFI, MF, N.

END OF TAPE PROCESSING



A. WRITE CASE

LABELLED TAPE	Y	N	Y	N
UP BIT SET	N	N	Y	Y
	1	2	3	4

B. READ CASE

LABELLED TAPE	Y	N	Y	N
UP BIT SET	N	N	Y	Y
	5	6	7	8

FET-FNT

Code & Status

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Record Level Number (bits 17–14)
Completion Status (bits 13–9)
Function Code (bits 8–2); Master Bits 4 3 → PP Read EOR-EOF Indicators
Binary-Coded Flag (bit 1)
Inactive-Active Flag (bit 0)

COMPLETION STATUS (BITS 9-13)

01 = EOI on For. or Back. Read
02 = End-of-Reel on For. or Back. Operation when up bit is set.
04 = Unrec. parity error
10 = Device of buffer capacity exceeded.
20 = Open function redundant
21 = Close function redundant
22 = Illegal function
23 = Index full
24 = FNT full

ADDITIONAL STACK

ORDER CODES

02 = Read-CM-Program (3 wrds lost)
10 = PP Read
11 = PP Overlay Read
14 = PP Write No EOR
15 = PP WRite EOR
17 = Release Chain

FUNCTION CODE (Bits 0—8)	Bits 2—8	Stack Order Code
*00X = Illegal	00-01	--
*010 = Read	02	00
*014 = Write	03	04
020 = Read-Skip	04	01
*024 = Write-EOR	05	05
*034 = Write EOF	07	05
*040 = Backspace	10	13
044 = Backspace PRU	11	16
*050 = Rewind	12	
*054 = Rewind	13	
*060 = Unload	14	
*064 = Unload	15	
100 = Open	20	
114 = Evict	114	
240 = Skip-For.	50	12
640 = Skip-Back.	150	13

INACTIVE-ACTIVE (BIT 0)
*0 = Active
*1 = Inactive

BINARY-CODED (BIT 1)
*0 = Coded
*1 = Binary

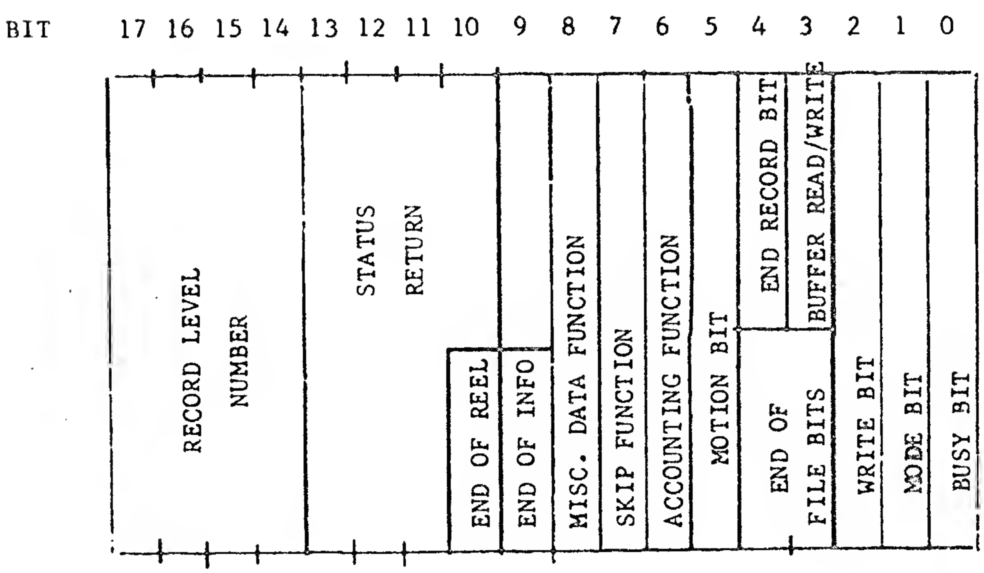
PP READ (BITS 3-4)
*10 = EOR Read
*11 EOF Read

* Identical Meaning in Scope 2.0

10-18

2.0/3.0 CIO operation code comparison

CODE	2.0 USAGE	3.0 USAGE	3.0 SYSTEM MACRO FORM	
00	NOT USED	READ PHYSICAL REC.	RPHR	LFN,RECALL
04	NOT USED	WRITE PHYSICAL REC.	WPHR	LFN,RECALL
10	READ	READ	READ	LFN,RECALL
14	WRITE	WRITE	WRITE	LFN,RECALL
20	NOT USED	READ SKIP	READSKP	LFN,1,RECALL
24	WRITE RECORD	WRITE RECORD	WRITER	LFN,1,RECALL
30	SEARCH FILE FORW.	ILLEGAL		
34	WRITE END FILE	WRITE END FILE	WRITEF	LFN,RECALL
40	BACKSPACE	BACKSPACE	BKSP	LFN,RECALL
44	BACKSPACE	BACKSPACE PHYSICAL	BKSPRU	LFN,N,RECALL
50	REWIND	REWIND	REWIND	LFN,RECALL
60	UNLOAD	UNLOAD	UNLOAD	LFN,RECALL
114	NOT USED		EVICT	LTN,RECALL
130	SEARCH FILE FORWARD	ILLEGAL		
240	NOT USED	SKIP FORWARD	SKIPF	LFN,N,1,RECALL
640	NOT USED	SKIP BACKWARDS	SKIPB	LFN,N,1,RECALL



LAST CODE AND STATUS
Found in FET(1) and FNT/FST(3)

BIT	USE		SYMPTOM
0	BUSY BIT	0	BUSY
		1	FREE
1	MODE BIT	0	CODED
		1	BINARY
2	WRITE BIT	0	READ
		1	WRITE
3,4	END OF FILE BITS	0	NO EOF
		3	EOF
4	END OF RECORD BIT	0	NO EOR
		1	EOR
5	MOTION BIT	0	FORWARD
		1	REVERSE
6	ACCOUNTING BIT	0	NO OPEN/CLOSE
		1	OPEN/CLOSE ACTION
7	SKIP FUNCTION BIT	0	NO SKIP
		1	SKIP
9	END OF INFO BIT	0	NO EOI
		1	EOI

STATUS-OPCODE BREAKDOWN-3.0 CIO CALL

10	END OF TAPE BIT	0	NO END OF TAPE
		1	END OF TAPE
9-13	STATUS RETURN	04	UNRECOVERABLE PARITY
		10	CAPACITY ERROR
		20	OPEN REDUNDANT
		21	CLOSE REDUNDANT
		22	ILLEGAL FUNCTION
		23	INDEX FULL
		24	FNT FULL

6000 PROGRAMMING

-EXAMPLES-

1. BASIC DECK STRUCTURE

2. SIMPLE FORTRAN JOB.

JOB CARD
REQUEST CARDS
COMPILE (OR ASSEMBLE) CARD(S)
EXECUTION CARDS
ERROR CONTROL CARDS
7-8-9
PROGRAM EXAMPLE (INPUT, OUTPUT)
.
.
.
END
7-8-9
DATA
7-8-9
6-7-8-9

SEE FOLLOWING EXAMPLES FOR FORTRAN AND ASSEMBLER RUNS.

FORTRAN CALL FOR A COMPASS SUBROUTINE

6000 SCOPE 3

12/18/67

COL 7

YOUR PROGRAM IN FORTRAN STATEMENTS

```
      CALL START (IFRØMW, ITØW, NUMW)
      STØP
      END
```

RETURN TO THIS STATEMENT

COL 7

PARAMETERS GO TO B1, B2, --- B6
IF # OF PARAMETERS IS > 6, SPACE MUST BE LEFT BEFORE HEADER WORD DEFINITION, BUT AFTER THE ENTRY STATEMENT.

MUST MATCH

COL 11

```
          IDENT   WMØVE
          ENTRY   START
          (BSSZ   X)
          VFD     42/5LWMØVE, 18/3
START     BSSZ    1
          ———     (1ST INSTRUCTION)
          ———
          ———
          ———
          ———     (LAST INSTRUCTION)
          EQ      START
          END
```

GENERATE HEADER WORD

RESERVE WORD FOR RETURN JUMP

YOUR SUBROUTINE IN COMPASS

UNCONDITIONAL JUMP TO THE RETURN JUMP

COL 11

IDENT AND END CARDS MUST BE IN COL 11, OTHERS DON'T HAVE TO BE.

DON'T USE A TRANSFER ADDRESS ON THE END CARD BECAUSE IT WOULD TRANSFER CONTROL TO SUBROUTINE AT "START" AT END OF COMPILE/ASSEMBLER.

Dec 16, 1967
14-Dec

to run a Compass program on 6000 SCOPE 3
without using the Fortran compiler —

```
            7
            8
            9     Col 11
                    ↓
(1st card) →       IDENT   NAME
                   ENTRY   MOVE    ← "PERIPH" CARD IF PROGRAM IS PP TYPE
           MOVE    (1st instruction)
                   —
                   —
                   —
                   |
                   |
                   |
                   |
                   —
                   —
                   —
                   (Last instruction) (NOT A "PS")
                   ENDRUN        (STORES "END" IN RA+1)
           AREAB   DATA 47 -- etc
                   —
                   —
                   —
                   —
                   —         } DATA, Common area, etc.
                   —
                   —
                   —
                   —
(Last card) →      END  MOVE   ← TERMINATE ASSEMBLER AND LOOKS FOR TRANSFER ADDRESS. (MUST HAVE)
            7
            8
            9
```

CORE DUMPS AND RELATED DEBUGGING AIDS

Robert W. Bartlett

Control Data Corporation

When a program terminates abnormally, the SCOPE monitoring system provides a one page dump of information consisting of:

1. The contents of all 24 operating registers
2. The contents of 7 core memory locations whose addresses are in registers A1 thru A7
3. A 200B word dump of the 100B words before and after the core address where the error was detected
4. A variable number of words beginning at location 000000 which contain system information, including the names and buffer parameter addresses of all files referenced by the program.

Only in rare instances will the average applications programer be able to make use of items (3) and (4) in the above list. Rather, he should typically make use of the following tools:

A. Items (1) and (2) above
B. A core map of routines, entry points, and references
C. A compilation listing of applicable routines

The concerted use of these tools should allow the programer to pinpoint the FORTRAN statement that was executing when the error occurred. In addition, it is often possible to determine the current value of several variables used in the statement. From then on, the programer's intimate knowledge of his own program should allow him to deduce what combination of faulty input, or coding, or logic caused this particular error to occur at this particular place. Let us now discuss in greater detail what can be found in these three debug tools.

A. One page dump (Figure 4)

The 24 operating registers consist of:

8 address registers (A0 thru A7) - 18 bits each
8 increment registers (B0 thru B7) - 18 bits each
8 operand registers (X0 thru X7) - 60 bits each

The contents of the A registers should correspond to addresses of memory locations within the field length of the job. An address in an A register equal to or greater than the field length specified on the job card would normally be associated with a mode 1 arith error.

To the right of the A and B register contents on the dump is a list of contents of memory locations specified by the associated A registers. For example, the contents of memory location 123B referenced in address register A3 are listed as C(A3)= 1721 6000 0000 0000 0000.

The contents of the X registers correspond to the values of operands involved in arithmetic or alpha-numeric operations. The programer's eye should be trained to recognize the following error indications in the C and X lists.

+ Indefinite	1777	----	----	----	----
+ Infinity	3777	----	----	----	----
- Infinity	4000	----	----	----	----
- Indefinite	6000	----	----	----	----

B. Core map (Figure 3)

From the core map of the attached sample program (figure 1), we can learn the following things which are true of core maps in general.

The first two lines tell us that the job loaded between addresses 100B and 2060B and that storage for BLANK COMMON begins at 2056B and runs for a length of 2 words.

The table headed - PROGRAM----ADDRESS - tells us that program DMPREAD began loading at address 103B and subroutine SYSTEM began at address 1143B. To the right of this table are the names and beginning addresses of the labeled common blocks associated with each routine. For example, the labeled common block named BLOK loaded at address 100B preceeding the program (DMPREAD) that first declared it.

The table headed --ENTRY-----ADDRESS- tells us that program DMPREAD has an entry point named DMPREAD at address 104B. The remaining eight entry points all belong to subroutine SYSTEM

since they all occur at locations greater than the loading address of SYSTEM at 1143B. They are:

Q8NTRY	at location	1144B
SYSTEM	at location	1307B
SYSTEMC	at location	1254B
SYSTEMP	at location	1302B
END	at location	1200B
STOP	at location	1227B
EXIT	at location	1221B
ABNORMAL	at location	1237B

In addition we notice that entry point Q8NTRY is referenced by location 105B found in DMPREAD and the entry point END is referenced by location 116B also found in DMPREAD. Since DMPREAD began loading location 103B, these calls are actually from relative address 2 and 13B in DMPREAD.

The table headed ---- UNSATISFIED EXTERNALS----- tells us that we forgot to load the subroutine SUB with this run. This subroutine is called from relative location 11B in DMPREAD, and the program would have quit on a mode 1 arith error at that point if other errors had not caused us to exit sooner.

C. Compilation listing (Figure 1)

The numbers along the left hand side of the compilation listing are the relative locations within the program at which the corresponding FORTRAN statements begin execution. For example, the set of machine language instructions necessary to perform the calculation AO = BO+CO**2 begins execution at relative location 6 in DMPREAD or at absolute location 111B in memory since DMPREAD begins at location 103B according to the core map.

On the page of information below the compilation listing is the following useful information. (Figure 2)

The list headed BLOCK NAMES AND LENGTHS tells us there are two common blocks associated with program DMPREAD. Block 01 is blank COMMON (the block name has been left blank) which is 2 words long, and block 02 is labeled COMMON which is 3 words long and is named BLOK.

The names and relative storage locations of variables used in program DMPREAD are:

AO at relative address 26B and absolute address 131B

C0 at relative address 25B and absolute address 130B

CO at relative address 01 in common block 02 and absolute address 101B

BO at relative address 01 in common block 01 and absolute address 2057B

BN, CN, and CP do not appear among the list of variables because they do not appear in any executable statement within the program.

Now that we know what can be found in core dumps, core maps, and compilation listings, let us see how they can be used together to find the source of error in the attached example. (Figure 1) But before we start let us list the three distinct arith errors that occur.

MODE1 Trying to reference an address that is equal to or greater than the field length.

MODE2 Trying to operate with an infinite operand (3777--- -------------or 4000----------------)

MODE4 Trying to operate with an indefinite operand (1777--- -------------or 4000----------------)

MODE3, MODE5, MODE6, MODE7 are algebraic combinations of these three i.e. MODE5 means MODE1 and MODE4.

Thus we will begin debugging by reading the DAYFILE message (figure 5) telling us that we have a MODE4 arith error at address 115B. Looking at the core map we see that program DMPREAD begins at location 103B and subroutine SYSTEM begins at address 1143B. Therefore, the error occurred in program DMPREAD and is in fact at relative address twelve in DMPREAD.

```
 000115B
-000103B
    =12B
```

If we now look at the listing for program DMPREAD, we see that location 12 corresponds on the processing of the END card. To escape from this dilemma, we must apply the following rule of thumb corrections to 6400 and 6600 addresses.

6600 Subtract 3 from error address
6400 Subtract 1 from error address

Then we find that the error occurred at relative address 7 while processing the FORTRAN statement AO=BO+CO**2.

```
 000012B
-000003B
     =7B
```

This is because the machine is looking ahead toward execution of future instructions before it discovers the error condition from the previous operation. Because of parallel operation the 6600 is looking further ahead than the serial operating 6400.

Now, knowing the logic of his program and the statement where the error showed up, the programer might be in a position to say to himself as follows:

> "The value of BO or CO must be indefinite since arith errors MODE1 thru 7 only occur during arithmetic operations on specious operands. But CO is equal to 3.0 and BO is equal to CO and both are good values. Oops - wait a minute there is a mispunch, BO = C0 and C0 is not previously defined on the left hand side of an arithmetic replacement statement. It is good that SETINDF (LRC programing manual, Q1.001) helped me to catch this mistake or there is no telling what answers I would have gotten on this or other machines."

In a longer program where the logic was not as easy to trace back, the debugging programer might go on to look at the register dumps for clues. Indeed, in our example we see that the contents of the memory locations referenced in address registers A1, A4, and A6 have the distinctive 1777 byte characteristic of indefinite operands.

```
C (A1) = Contents of 002057 = 1777 0000 0000 0000  0130B
C(A4) = Contents  of 000130 = 1777 0000 0000 0000  0130B
C(A6) = Contents of  000131 = 1777 0000 0000 0000  0000B
```

Making use of the core map and doing the familiar octal subtractions, we find that A4 and A6 are referencing the 25th and 26th locations in DMPREAD, and A1 is referencing the 01 element in BLANK COMMON.

A4	A6	A1
000130B	000131B	002057B
-000103B	-000 103B	002056B
=25B	=26B	=1B

Then looking at the list of variable assignments on the page below the compilation listing for DMPREAD, we see that

A1 is referencing BO - 000001C01
A4 is referencing C0 - 000025
A6 is referencing AO - 000026

By some thunderbolt technique we should soon discover that C0 is unwanted and undefined.

Final mention should be made of the use of a library routine called SETINDF as an aid in debugging this example. The control card SETINDF. placed after the RUN(S) and before LGO. Writes the indefinite indicator 1777 into the upper byte of every memory location from address 100B up to your field length. In addition, it writes each memory location's own address into the bottom half of each word. For example, core address 200B contains 1777 0000 0000 0000 0200B. Thus, when your program references a variable that has not been defined, the preset indefinite value will be used and should soon cause a MODE4 arith error to occur. An indefinite value caused by arithmetic operations of the machine will look like 1777 0000 0000 0000 0000B, i.e. the 1777 indefinite flag will be followed by all zeroes to the right. Thus, the telltale 130 in C(A4) and C(A1) should have sent us immediately looking for the undefined variable C0.

```
              PROGRAM DMPREAD(OUTPUT=1001)
000003        COMMON BN , BO
000003        COMMON /BLOK/ CN , CO , CP
000003        CO = 3.0
000005        BO = CO
000006        AO = BO+CO**2
000011        CALL SUB
000012        END
```


Figure 1 Compilation Listing of Sample Program

```
PROGRAM LENGTH INCLUDING I/O BUFFERS
001040
FUNCTION ASSIGNMENTS

STATEMENT ASSIGNMENTS

BLOCK NAMES AND LENGTHS
      -  000002    BLOK    -  000003
VARIABLE ASSIGNMENTS
AO    -  000026    BO      -  000001C01 CO      -  000001C02 CO      -  000025
START OF CONSTANTS
000014
START OF TEMPORARIES
000022
START OF INDIRECTS
000025
UNUSED COMPILER SPACE
006600
```

Figure 2 Program Information

8-14

```
CORE MAP  11.42.26.  NORMAL              CONTROL                                                    000100    002060    002056    000002
          ---TIME---LOAD MODE --L1--L2-----TYPE---------------USER-------CALL-------------FWA LOAD--LWA LOAD--BLNK COMN--LENGTH--
          FWA LOADER  035045  FWA TABLES  034771
          -PROGRAM-----ADDRESS-                           --LABELED---COMMON--
          DMPREAD     000103                                        BLOK        000100
          SYSTEM      001143
          --ENTRY------ADDRESS-                                                 REFERENCES
          DMPREAD     000104
          Q8NTRY      001144           DMPREAD      000105

          SYSTEM      001307

          SYSTEMC     001254
          SYSTEMP     001302
          END         001200           DMPREAD      000116

          STOP        001227

          EXIT        001221
          ABNORML     001237
          ----UNSATISFIED EXTERNALS-----                                        REFERENCES
          SUB                                       000114
```


Figure 3 Core Map

H-9

```
DMPX.

 P    000000   A0  040000   B0  000000
 RA   073500   A1  002057   B1  000103   C(A1)=  1777  0000  0000  0000  0130
 FL   040000   A2  000117   B2  000002   C(A2)=  0000  0000  0000  0000  0027
 EM   070000   A3  000123   B3  001143   C(A3)=  1721  6000  0000  0000  0000
               A4  000130   B4  000000   C(A4)=  1777  0000  0000  0000  0130
               A5  000101   B5  035227   C(A5)=  1721  6000  0000  0000  0000
               A6  000131   B6  000000   C(A6)=  1777  0000  0000  0000  0000
               A7  000136   B7  000001   C(A7)=  0000  0000  0000  0000  1143
 X0  7777  7777  7777  7700  0000
 X1  1777  0000  0000  0000  0130
 X2  1723  4400  0000  0000  0000
 X3  1721  6000  0000  0000  0000
 X4  1777  0000  0000  0000  0130
 X5  1721  6000  0000  0000  0000
 X6  1777  0000  0000  0000  0000
 X7  1777  0000  0000  0000  0000
000016   00000 00000 00000 00000   000021*14071 70000 00000 00047          00000 00000 00000 22676  000025*00000 00000 00000 32357
000024   00000 00000 00000 00000   000031*11162 02524 00000 00021          00000 00000 00000 36767         00000 00000 00000 37015
000035   00000 00000 00000 40000          00000 00000 00000 00000   000055*00000 00000 00000 22676.        00000 00000 00000 00002
000057   00000 00000 00000 32300
000060   00000 00000 00000 00003   000062*00000 00000 00000 00000          00000 00000 00000 40000
000064   14071 70000 00000 00000          00000 00000 00000 02060          00034 77100 00000 00103         00000 00000 60000 35045
000070   14071 75700 00000 00000          05301 12457 55000 00000          22052 12505 23245 52401         20054 15610 11575 55522
000074   11145 62301 26052 42056          03040 35522 05051 45516          00000 00000 00000 40000         00000 00000 00000 00000
000100   17770 00000 00000 00100          17216 00000 00000 00000          17710 00000 00000 00102         04152 02205 01040 00101
000104   61100 00103 61200 00117          01000 01144 30000 00000          51100 00121 10630 46000         51400 00101 51400 00130
000110   10640 51600 02057 46000          51500 00101 40255 46000          51100 00057 30712 24607         51400 00131 46000 46000
000114   01000 40000 07000 00103          51300 00124 10730 46000          01500 01700 07000 00103         00000 00000 00000 00027
000120   00000 00000 00000 00000          00000 00000 00000 10000          [illegible] 42025 24000 [illegible]         17216 00000 00000 00000
000124   00000 00000 00000 00000   000126*17770 00000 30000 00126          17770 00000 00000 00127
000130   17770 00000 00000 00130          17770 00000 00000 00000          17252 42025 24000 00135         00000 00000 00000 00142
000136   00000 00000 00000 01143          00000 00000 00000 00000   000141*00000 00000 00000 10000         17770 00000 00000 00142
000143   17770 00000 00000 00143
000144   17770 00000 00000 00144          17770 00000 00000 00145          17770 00000 00000 00146         17770 00000 00000 00147
000150   17770 00000 00000 00150          17770 00000 00000 00151          17770 00000 00000 00152         17770 00000 00000 00153
000154   17770 00000 00000 00154          17770 00000 00000 00155          17770 00000 00000 00156         17770 00000 00000 00157
000160   17770 00000 00000 00160          17770 00000 00000 00161          17770 00000 00000 00162         17770 00000 00000 00163
000164   17770 00000 00000 00164          17770 00000 00000 00165          17770 00000 00000 00166         17770 00000 00000 00167
000170   17770 00000 00000 00170          17770 00000 00000 00171          17770 00000 00000 00172         17770 00000 00000 00173
000174   17770 00000 00000 00174          17770 00000 00000 00175          17770 00000 00000 00176         17770 00000 00000 00177
000200   17770 00000 00000 00200          17770 00000 00000 00201          17770 00000 00000 00202         17770 00000 00000 00203
000204   17770 00000 00000 00204          17770 00000 00000 00205          17770 00000 00000 00206         17770 00000 00000 00207
000210   17770 00000 00000 00210          17770 00000 00000 00211          17770 00000 00000 00212         17770 00000 00000 00213
000000   00040 00115 00000 00000          00000 00000 00000 00000          17252 42025 24000 00132         00000 00000 00000 00000
```

Figure 4 Core Dump

```
14.35.41. PTB1967. READ.
14.35.42. PTB1967. PP 000 SEC.
15.00.06. PTB1967. PTB1967.
15.00.06. PTB1967.                       LRC COMPUTER COMPLEX
15.00.06. PTB1967.
15.00.06. PTB1967. JOB,01,100,40000.           P9970,12234,2,
15.00.06. PTB1967. R W BARTLETT             ,  CDC       ,
15.00.07. PTB1967. RUN(S)
15.00.11. PTB1967. SETINDF.
15.00.11. PTB1967. LGO.
15.00.13. PTB1967. ARITH ERROR.
15.00.13. PTB1967. MODE = 4. ADDRESS = 000115
15.00.13. PTB1967. CP  000.106 SEC.
15.00.13. PTB1967. PP  002.453 SEC.
15.00.22. PTB1967. PRINT-PP 000 SEC.
06/01/67. - SCOPE 2.0 - LRC 6600 - B -05/17/67.
```

14-11

Figure 5 Dayfile

```
VER 1           SETINDF                                         061067L006

                                                  IDENT         SETINDF
                          000C13        PRCGRAM LENGTH

                                        BLCCKS

                    CCC000    000C13    PRCGRAM*    LCCAL

                                        ENTPY POINTS

                                           000006 SETINDF

                                                  ENTRY         SETINDF
     000000   56620                     SET       SA6           B2
                    36661                         IX6           X6+X1
                         66221                    SB2           B2+B1
     000001   C7230CCC00 +                        LT            B2,B3,SET
                         7170C51604               SX7           51604B
     000002   20752                               LX7           42
                    56710                         SA7           B1
                         10644                    BX6           X4
                              54640               SA6           A4
     000003   36661                               IX6           X6+X1
                    54661                         SA6           A6+B1
                         36661                    IX6           X6+X1
                              54661               SA6           A6+B1
     000004   36661                               IX6           X6+X1
                    54661                         SA6           A6+B1
                         36661                    IX6           X6+X1
                              54661               SA6           A6+B1
     000005   23052411160406CCC000                VFD       42/0HSETINDF,18/0
     000006   46000                     SETINDF   NO
                    6110C00001                    SB1           1
                              76110               SX1           B1
     000007   6120000005 +                        SB2           SETINDF-1
                         76220                    SX2           B2
                              64300               SB3           A0
     000010   5140CCC000 +                        SA4           SET
                         74040                    SX0           A4
     000011   7140001777                          SX4           17773
                         20460                    LX4           48
                              36642               IX6           X4+X2
     000012   36440                               IX4           X4+X0
                    C200000000 +                  JP            SET
     000C13                                       END           SETINDF
```

6000 SCOPE 3

CREATE EDITSYM LISTING OF A SIMPLE PROGRAM

PAGE-1 →

TEST006

PAGE-2 →

TEST006

PAGE-3 IS BLANK

PAGE-4 →

EDITSYM CONTROL CARDS

*DECK,TRYIT

PAGE-5 THRU 8 ARE THE FOLLOWING PAGES.

PAGE-9 IS BLANK

PAGE-10 IS THE "IDENT TRY COMP" PAGE

PAGE-12 IS THE "CORE MAP"

PAGE-13 IS THE "DMPX"

PAGE-14 IS THE "DAYFILE"

EDITSYM LIST   PAGE NO.   00001

```
      PROGRAM TRYIT(INPUT,OUTPUT)                                   TRYIT00001
      COMMON I,J,K,L,M                                              TRYIT00002
      READ 10,I,J,K,L                                               TRYIT00003
   10 FORMAT(4(I5,/))                                               TRYIT00004
      CALL TEST(I,J,K,L,M)                                          TRYIT00005
      PRINT 11,M                                                    TRYIT00006
   11 FORMAT(1H1,I7)                                                TRYIT00007
      STOP                                                          TRYIT00008
      END                                                           TRYIT00009
           IDENT TRYCOMP                                            TRYIT00010
           ENTRY TEST                                               TRYIT00011
           LIST L,R                                                 TRYIT00012
           VFD 42/7LTRYCOMP,18/5      .GENERATE THE HES             TRYIT00013
 TEST      BSSZ 1         .RESERVE A WORD FOR THR RETURN JUMP INSTRUCTION TRYIT00014
           SA1 B1         .LOAD THE FIRST CONSTANT FRON I (B1)      TRYIT00015
           SA2 B2                                                   TRYIT00016
           SA3 B3                                                   TRYIT00017
           SA4 B4                                                   TRYIT00018
           IX5 X1-X2      .DIFF OF I AND J TO X5                    TRYIT00019
           ZR X5,ADD      .JUMP TO K+L IF X5 IS ZERO                TRYIT00020
           PL X5,SUB      .JUMP TO K-L IF X5 IS POSITIVE            TRYIT00021
           IX6 X4-X3      .SUB L-K                                  TRYIT00022
           EQ STORE                                                 TRYIT00023
 ADD       IX6 X3+X4      .ADD K TO L                               TRYIT00024
           EQ STORE                                                 TRYIT00025
 SUB       IX6 X3-X4      .SUBTRACT K FROM L                        TRYIT00026
 STORE     SA6 B5                                                   TRYIT00027
           EQ TEST        .RETURN TO THE RETURN JUMP INSTRUCTION    TRYIT00028
           END                                                      TRYIT00029
```

```
               PROGRAM TRYIT(INPUT,OUTPUT)                           TRYIT00001
 000000  0              L00002
 000001  6110 L00002    L00001
         6120 C00001
 000002  0100 S00100
               COMMON I,J,K,L,M                                      TRYIT00002
               READ 10,I,J,K,L                                       TRYIT00003
 000003  6130 N00001
         6120 C00006
 000004  67202
         66100
 000005  0100 S00200    L00005
         0700 L00002
 000006  66200
         6110 V00001
 000007  0100 S00200
 000010  66200
         6110 V00002
 000011  0100 S00200
 000012  66200
         6110 V00003
 000013  0100 S00200
 000014  66200
         6110 V00004
 000015  0100 S00200
 000016  6110 77777б
         0100 S00200
            10 FORMAT(4(I5,/))                                       TRYIT00004
               CALL TEST(I,J,K,L,M)                                  TRYIT00005
 000017  6110 V00001
         6120 V00002
 000020  6130 V00003
         6140 V00004
 000021  6150 V00005
 000022  0100 S00300    L00007
         0705 L00002
               PRINT 11,M                                            TRYIT00006
 000023  6130 N00002
         6120 C00012
 000024  67202
         66100
 000025  0100 S00400    L00012
         0700 L00002
 000026  66200
         6110 V00005
 000027  0100 S00400
 000030  6110 77777б
         0100 S00400
            11 FORMAT(1H1,I7)                                        TRYIT00007
```

```
STOP                                            TRYIT00008
000031  5130 C00015
        10733
000032  0100 S00500   L00014
        0700 L00002
        END                                     TRYIT00009
000033  5140 C00016
        10744
000034  0100 S00600   L00015
        0700 L00002
```

```
PROGRAM LENGTH INCLUDING I/O BUFFERS
004121

FUNCTION ASSIGNMENTS

STATEMENT ASSIGNMENTS
10       =   000044    11       =   000050

BLOCK NAMES AND LENGTHS
         =   000005

VARIABLE ASSIGNMENTS
I        =   000000C01  J       =   000001C01  K       =   000002C01  L       =   000003C01
M        =   000004C01

START OF CONSTANTS
000035

START OF TEMPORARIES
000053

START OF INDIRECTS
000055

UNUSED COMPILER SPACE
043600
```

```
                                                IDENT TRYCOMP                                                   TRYITO
                        000011    PROGRAM LENGTH

                                  BLOCKS

                000000  000011    PROGRAM*    LOCAL

                                  ENTRY POINTS

                                     000001 TEST

                                                 ENTRY TEST                                                     TRYIT00011
                                                 LIST L,R                                                       TRYIT00012
  000000  24223103171520000005                   VFD 42/7LTRYCOMP,18/5      .GENERATE THE HES                   TRYIT00013
  000001                              TEST       BSSZ 1            .RESERVE A WORD FOR THR RETURN JUMP INSTRUCTION  TRYIT00014
  000002  56110                                  SA1 B1            .LOAD THE FIRST CONSTANT FRON I (B1)         TRYIT00015
               56220                             SA2 B2                                                         TRYIT00016
                    56330                        SA3 B3                                                         TRYIT00017
                         56440                   SA4 B4                                                         TRYIT00018
  000003  37512                                  IX5 X1-X2         .DIFF OF I AND J TO X5                       TRYIT00019
               0305000006 +                      ZR X5,ADD         .JUMP TO K+L IF X5 IS ZERO                   TRYIT00020
  000004  0325000007 +                           PL X5,SUB         .JUMP TO K-L IF X5 IS POSITIVE               TRYIT00021
               37643                             IX6 X4-X3         .SUB L-K                                     TRYIT00022
  000005  0400000010 +                           EQ STORE                                                       TRYIT00023
  000006  36634                       ADD        IX6 X3+X4         .ADD K TO L                                  TRYIT00024
               0400000010 +                      EQ STORE                                                       TRYIT00025
  000007  37634                       SUB        IX6 X3-X4         .SUBTRACT K FROM L                           TRYIT00026
  000010  56650                       STORE      SA6 B5                                                         TRYIT00027
               0400000001 +                      EQ TEST           .RETURN TO THE RETURN JUMP INSTRUCTION       TRYIT00028
  000011                                         END                                                            TRYIT00029

                        053330    UNUSED STORAGE           20 STATEMENTS        4  SYMBOLS
```

SYMBOLIC REFERENCE TABLE

ADD	0000006	PROGRAM*	000003	
STORE	0000010	PROGRAM*	000005,	000006
SUB	0000007	PROGRAM*	000004	
TEST	0000001	PROGRAM*	000010	

```
CORE MAP *00.37.38.  NORMAL                CONTROL                                                000100     010513     010506     000005
---TIME---LOAD MODE --L1--L2-----TYPE---------------USER--------CALL-------------FWA LOAD--LWA LOAD--BLNK COMN--LENGTH--
          FWA LOADER  074614  FWA TABLES  074403
          -PROGRAM----ADDRESS-                      --LABELED---COMMON--
          TRYIT       000100
          TRYCOMP     004221
          SYSTEM      004232                                  SCOPE2      004232
          INPUTC      005207
          OUTPTC      006306
          SIO$        007640
          GETBA       010467
          --ENTRY----ADDRESS-                                            REFERENCES
          TRYIT       000101
          TEST        004222           TRYIT     000122

          Q8NTRY      004233           TRYIT     000102

          SYSTEM      004433           INPUTC    005246    006117
                                       OUTPTC    006322    007451

          SYSTEMC     004400
          SYSTEMP     004426
          END         004324           TRYIT     000134

          STOP        004353           TRYIT     000132

          EXIT        004345
          ABNORML     004363           INPUTC    005247    006120
                                       OUTPTC    006323    007452

          INPUTC      005211           TRYIT     000105    000107    000111    000113    000115    000116

          KRAKER      005313
          OUTPTC      006310           TRYIT     000125    000127    000130

          KODER       006444
          BKSPRU.     010136
          FIZHAK.     010147
          POSFIL.     010205           OUTPTC    006340

          RDPRU.      010215
          DAT.        010237           INPUTC    005265    005233    005256
                                       OUTPTC    006362    006376    006343

          CIO1.       010103
          OPEN.       007642           SYSTEM    004704
                                       INPUTC    005231
                                       OUTPTC    006330

          SIO.        007755           INPUTC    005261
                                       OUTPTC    006375

          GETBA       010467           INPUTC    005217
                                       OUTPTC    006316

          ----UNSATISFIED EXTERNALS-----                                 REFERENCES
```

PAGE - 13

14-21

```
DMPX.

P   000000   A0  000000   B0  000000
RA  123200   A1  006271   B1  010475   C(A1)=  0000  0000  0000  0000  0000
FL  100000   A2  000077   B2  010476   C(A2)=  0000  0000  0000  0000  0000
EM  070000   A3  005263   B3  010477   C(A3)=  1116  2025  2400  0000  0000
             A4  000156   B4  010500   C(A4)=  0203  0000  0000  1000  0176
             A5  000173   B5  010501   C(A5)=  7777  7777  7777  7777  7777
             A6  006271   B6  000001   C(A6)=  0000  0000  0000  0000  0000
             A7  000162   B7  006252   C(A7)=  0000  0000  0000  0000  0004
X0  7777  7777  7777  7777  7773
X1  0000  0000  0000  0000  0000
X2  0000  0000  0000  0000  0000
X3  1116  2025  2400  0000  0000
X4  7777  7777  7777  7700  0000
X5  0000  0000  0000  0000  0000
X6  0000  0000  0000  0000  0000
X7  0000  0000  0000  0000  0004
000000   00011 00001 00000 00000      000001= 00000 00000 00000 00000
000004   00000 00000 00000 00000      000006→17252 42025 24000 00120
000010   00000 00000 00000 10000      000011= 00000 00000 00000 00000
000016   00000 00000 00000 00000      000020→04000 12227 00000 00000
000023   00000 00000 00000 31056      000025→22560 62000 01000 75410
000043   15112 32311 16070 00000
000044   55555 55555 55555 55555      000050→55555 55524 22311 12433
000053   23312 32405 30240 00000
000054   00000 00000 00000 00000      000055= 00000 00000 00000 31027
000060   00000 00000 00000 00004      000062→00000 00000 00000 00000
000064   14071 70000 00000 00000      000065= 00000 00000 00000 10502
000070   14071 75700 00000 00000      000071= 00000 00000 00000 00000

000002= 11162 02524 00000 00155      000003= 17252 42025 24000 02177
000007= 00000 00000 00000 00006
000014→00000 00000 00001 00000       000015= 14071 70000 00000 00000
000021= 14071 70000 00000 00047      000022= 02030 00000 00100 31027
000026= 00000 00000 00000 00000      000042→55051 60455 03012 20455

000051= 33333 54455 55555 50000      000052= 00000 00000 00000 00000

000056= 00000 00000 00000 00002      000057= 00000 00000 00000 75277
000063= 00000 00000 00001 00000
000066= 00074 40700 00000 00100      000067= 00000 00000 60000 74614
000076→00000 00000 00001 00000
```

PAGE - 14

```
02/05/68  SCOPE V3.1 6500-00 131K 11/21/67
00.50.49.TEST006
00.50.49.TEST,P17,T100,CM100000.
00.50.49.EDITSYM(I,C,L,,NPL)      .GENERATE AN EDITSYM LISTING OF THE
00.50.49.PROGRAM
00.50.52.RUN(M,,,COMPILE)
00.51.03.LGO.
00.51.06.ARITH ERROR.
00.51.06.MODE = 1.   ADDRESS =100001
00.51.06.CP   005.159 SEC.
00.51.06.PP   014.537 SEC.
```

the following listing illustrates the results of using UPDATE to obtain an assembly of a system program for later modifications.

the 1st Page is -

SCPD01F

the 2nd Page is -

SCPD01F

the 3rd page is blank

the following pages are the 4th and on.

```
#IDENT,RAK
#COMPILE.MSG.
```

4-213

COMPILE	1SP1SQ	FNTSRCH	READPP	BUFINIT	RELOC
PPTIME	COMCOMP	TRNCOM	TAPEDS	DISCUS	DUMPDS
RECOV	PREPP	PRECP	STLPP	STLCP	CMR
MTR	DSD	CIO	LDR	MSG	OPE
1AJ	1BJ	ILJ	1DT	1SP	1SQ
2BP	2DF	2ES	2TR	2TS	2TW
7SD	ATS	CHK	CKP	CLL	CLO
CLS	CPL	CTS	DIS	DMP	EXU
LBC	LOC	LOD	MDI	MEM	PBC
RBR	REQ	RFL	RST	SRB	TIM
WBR	1BT	1CO	1CY	1DF	1EE
1LT	1MF	1MR	1MW	1PL	1PO
1RI	1RO	1SX	1TD	2BT	2CA
2CF	2CJ	2EF	2LA	2LB	2LE
2LP	2PC	2PC	2RT	2TB	2TF
2TJ	2WT	3DT	4LB	5DT	7TP
LOADER	OVERLOD	QXX	EDITLIB	EDITSYM	COPYBCD
COPYL	CPC	IORANDOM	IO	COMPARE	RESTART
BKSP	COPY	COPYBF	COPYN	COPYSBF	REWIND
UPDATE	CVRT	ACGOER	DBLE	EXP	GETBA
IBAIEX	LEGVAR	LOCF	SINCOS	SNGL	SQRT
SYSTEM	TAN	XRCL	ALNLOG	ASINCOS	ATAN
ATAN2	BACKSP	BUFFEI	BUFFEO	CABS	CBAIEX
CCOS	CEXP	CLOG	CSIN	CSQRT	DABS
DATAN	DBADEX	DBAIEX	DEXP	DISPLA	DLNLOG
DMOD	DSIGN	DSINCOS	DSQRT	DUMP	DVCHK
ENDFIL	IDINT	IFENDF	INPUTB	INPUTC	INPUTN
INPUTS	IOCHEC	IOCHEK	LENGTH	OUTPTB	OUTPTC
OUTPTN	OUTPTS	OVERFL	OVERLAY	PAUSE	RANF
RBAIEX	RBAREX	REMARK	REWINM	SCOPE2B	SECOND
SEGMENT	SIO$	SLITE	SLITET	SSWTCH	START
TANH	TIME	COMPASS	CVDEL	RUN	RUN1
QBDIAGP	TRANS	SYSTEXT	SCOP31A	SCOP31B	IPARAMS
SCPTEXT	SCOP31C	SCOP31E	SCOP31D	SERLO	87S0019
86C0186	86S0351	86S0204	86S0248	86S0257	86S0259
86S0275	86S0297	86S0393	86S0412	87S0445	86S0449
87S0480	87S0481	87S0482	87S0483	87S0524	87S0525
87S0527	86S0528	86S0529	86S0530	87S0531	86S0532
86S0533	86C0534	86F0535	91S0025	91S0041	89S0123
89S0253	89S0273	89S0365	91S0366	91S0402	89S0432
89S0446	87S0456	89S0458	89S0485	86S0495	89S0541
89S0550	87S0552	90S0582	89S0583	87S0588	87S0589
87S0590	87S0592	89S0595	89S0597	89S0598	89S0605
89S0607	89S0614	91S0664	91S0666	91S0667	91S0668
91S0669	91S0671	91S0672	91S0674	91S0675	91S0676
87F0333	87F0403	90F0466	86C0214	86C0217	86C0237
86C0244	86C0263	86C0281	86C0314	87C0490	87C0567
87C0568	87C0569	87C0580	91C0652	91C0663	91C0665
94S0747	SYSALT	91S0670	RAN		

114-24

DECK NAMES CURRENTLY KNOWN 02/08/68 PAGE NO. 00003

COMPILE	1SP1SQ	ENTSRCH	READPP	BUFINIT	RELOC
PPTIME	COMCOMP	TRNCOM	TAPEDS	DISCDS	DUMPDS
RECOV	PREPP	PRECP	STLPP	STLCP	CMR
MTR	DSD	CIO	LDR	MSG	OPE
1AJ	1BJ	1LJ	1OT	1SP	1SQ
2BP	2DF	2ES	2TR	2TS	2TW
7SB	ATS	CHK	CKP	CLL	CLO
CLS	CPL	CTS	DIS	DMP	EXU
LRC	LOC	LOD	MDI	MEM	PBC
RBR	REQ	RFL	RST	SRB	TIM
WBR	1BT	1CO	1CY	1DF	1EE
1LT	1MF	1MR	1MW	1PL	1PO
1RI	1RO	1SX	1TD	2BT	2CA
2CE	2CJ	2EF	2LA	2LB	2LE
2LP	2PC	2RC	2RT	2TB	2TF
2TJ	2WT	3OT	4LB	5DT	7TP
LOADER	OVERLOD	QXX	EDITLIB	EDITSYM	COPYBCD
COPYL	CPC	IORANDM	IO	COMPARE	RESTART
BKSP	COPY	COPYBF	COPYN	COPYSBF	REWIND
UPDATE	CVRT	ACGOER	DBLE	EXP	GETBA
IBAIEX	LEGVAR	LOCF	SINCOS	SNGL	SQRT
SYSTEM	TAN	XRCL	ALNLOG	ASINCOS	ATAN
ATAN2	BACKSP	BUFFEI	BUFFEO	CABS	CBAIEX
CCOS	CEXP	CLOG	CSIN	CSQRT	DABS
DATAN	DBADEX	DBAIEX	DEXP	DISPLA	DLNLOG
DMOD	DSIGN	DSINCOS	DSQRT	DUMP	DVCHK
ENDFIL	IDINT	IFENDF	INPUTB	INPUTC	INPUTN
INPUTS	IOCHEC	IOCHEK	LENGTH	OUTPTB	OUTPTC
OUTPTN	OUTPTS	OVERFL	OVERLAY	PAUSE	RANF
RBAIEX	RBAREX	REMARK	REWINM	SCOPE2B	SECOND
SEGMENT	SIO$	SLITE	SLITET	SSWTCH	START
TANH	TIME	COMPASS	CVDEL	RUN	RUN1
QBD1AGP	TRANS	SYSTEXT	IPARAMS	SCPTEXT	DSLCOM
CPSYS					

14-25

*WEOR-S, *CALLS-S AND MODIFICATIONS 02/08/68 PAGE NO. 00004

*YANK,SCOP31E SERLO 00001

1/4-2-2

COMMON DECKS ENCOUNTERED 02/08/68 PAGE NO. 00005

```
DSLCOM   IPARAMS   ISPISQ    FNTSRCH   READPP   BUFINIT
RELOC    PPTIME    COMCOMP   TRNCOM
```

1/4-27

LIST OF DECKS WRITTEN ONTO COMPILE FILE 02/08/68 PAGE NO. 00006

MSG

```
                                        IDENT  MSG,C,PPFWA                                              MSG     00002
                      001157    PROGRAM LENGTH
                                BLOCKS
             000000   001157    PROGRAM*   ABSOLUTE
                                        PERIPH                                                          MSG     00003
                                *                                                                       MSG     00005
                                *       .         ***FUNCTION***                                        MSG     00006
                                *                                                                       MSG     00007
                                *       .MSG IS USED TO ENTER MESSAGES FROM A CENTRAL MEMORY PROGRAM    MSG     00008
                                *       .INTO THE DAYFILE                                               MSG     00009
                                *                                                                       MSG     00010
                                *                                                                       MSG     00011
                                *                                                                       MSG     00012
                                *       .         ***REVISIONS***                                       MSG     00013
                                *                                                                       MSG     00014
                                *       .REVISION 1.                                                    MSG     00015
                                *       .         8/2/66. TO PUT THE DAYFILE MESSAGE COUNTER IN         MSG     00016
                                *       .         MSG INSTEAD OF MTR.                                   MSG     00017
                                *       REVISION 2. (M. C. STEELE)                                      MSG     00018
                                *                 CONVERSION TO SCOPE 3.0                               MSG     00019
                                *                                                                       MSG     00020
                                *       .         ***ENTRY AND EXIT INFORMATION***                      MSG     00021
                                *       .ENTRY                                                          MSG     00022
                                *       .         BITS 0-17 OF RA+1 SPECIFY THE ADDRESS OF THE MESSAGE  MSG     00023
                                *       .         TO BE ENTERED. BITS 24-35 SPECIFY WHETHER THE         MSG     00024
                                *       .         MESSAGE WILL BE ENTERED IN THE DAYFILE OR MERELY      MSG     00025
                                *       .         DISPLAYED ON THE SCOPE. IF THESE BITS ARE ZERO,MESS.  MSG     00026
                                *       .         WILL BE ENTERED IN DAYFILE  NONZERO-JUST DISPLAYED.   MSG     00027
                                *                                                                       MSG     00028
                                *       .EXIT                                                           MSG     00029
                                *       .         NONE                                                  MSG     00030
                                *                                                                       MSG     00031
                                *                                                                       MSG     00032
                                *       .         ***REGISTER AND BUFFER FORMATTING***                  MSG     00033
                                *                                                                       MSG     00034
                                *       .BITS 42-59 OF IR SPECIFY THE PROGRAM NAME,BITS 36-38 SPECIFY   MSG     00035
                                *       .CONTROL POINT,BITS 24-35 SPECIFY WHETHER MESSAGE WILL BE       MSG     00036
                                *       .ENTERED INTO DAYFILE OR JUST DISPLAYED,BITS 0-17 SPECIFY       MSG     00037
                                *       .ADDRESS OF MESSAGE.                                            MSG     00038
                                *                                                                       MSG     00039
                                *       .MSG PLACES THE RECEIVED MESSAGE INTO THE MESSAGE BUFFER        MSG     00040
                                *                                                                       MSG     00041
                                *                                                                       MSG     00042
                                *                                                                       MSG     00043
                                        SST                                                             SCOP31C 00260
```

```
      1000                       ORG    C.PPFWA                                   MSG      00046
      1000   1401                LDN    1                                         SCOP31A  00419
      1001   3470                STD    D.PPONE                                   SCOP31A  00420
      1002   3075                LDD    D.PPIR                                    SCOP31A  00421
      1003   6050                CRD    D.PPIRB       READ INPUT REGISTER         SCOP31A  00422
      1004   3053                LDD    D.PPIRB+3     OBTAIN ARGUMENT ADDRESS     MSG      00048
      1005   1237                LPN    37B                                       SCOP31A  00423
      1006   3464                STD    D.OUT                                     MSG      00049
      1007   1014                SHN    12                                        MSG      00050
      1010   3154                ADD    D.PPIRB+4                                 MSG      00051
      1011   3465                STD    D.OUT+1                                   MSG      00052
      1012   0200 0634           RJM    R.TFL         COMPARE TO FIELD LENGTH     MSG      00053
      1014   0732                MJN    MS3           SENSE ARGUMENT OUT OF BOUNDS SCOP31A 00424
      1015   6040                CRD    D.BA          READ POSSIBLE INDIRECT ARGUMENT SCOP31A 00425
      1016   3051                LDD    D.PPIRB+1                                 MSG      00055
      1017   1220                LPN    20B                                       MSG      00056
      1020   3462                STD    D.IN          RECALL BIT                  MSG      00057
      1021   0412                ZJN    MSGA          NO RECALL,ARGUMENT DIRECT   MSG      00058
      1022   3042                LDD    D.BA+2                                    MSG      00065
      1023   1377                SCN    77B                                       MSG      00066
      1024   1006                SHN    6                                         MSG      00067
      1025   3341                LMD    D.BA+1                                    MSG      00068
      1026   1006                SHN    6                                         MSG      00069
      1027   3465                STD    D.OUT+1       REAL MESSAGE ADDRESS        MSG      00070
      1030   1063                SHN    -12                                       MSG      00071
      1031   1237                LPN    37B                                       SCOP31A  00426
      1032   3464                STD    D.OUT                                     MSG      00072
      1033   1410          MSGA  LDN    8                                         SCOP31A  00427
      1034   3401                STD    D.Z1          CM WORD COUNTER             SCOP31A  00428
      1035   3404                STD    D.Z4                                      SCOP31A  00429
      1036   2000 1155           LDC    BUF-1                                     SCOP31A  00430
      1040   3402                STD    D.Z2          LOCAL BUFFER POINTER        SCOP31A  00431
      1041   3064          MS1   LDD    D.OUT         PICK UP A WORD OF THE MESSAGE MSG    00079
      1042   1014                SHN    12                                        SCOP31A  00432
      1043   3165                ADD    D.OUT+1                                   MSG      00083
      1044   0200 0634           RJM    R.TFL                                     SCOP31A  00433
      1046   0732          MS3   MJN    MS4           SENSE ADDRESS OUT OF RANGE (STEP) SCOP31A 00434
      1047   6170 1156     MS3A  CRM    BUF,D.PPONE   READ NEXT CM WORD OF MESSAGE SCOP31A 00435
      1051   1405                LDN    5                                         SCOP31A  00436
      1052   3403                STD    D.Z3          BYTE COUNTER                SCOP31A  00437
      1053   5500 1050           RAM    MS3A+1        ADVANCE BUFFER READ ADDRESS SCOP31A  00438
      1055   3002          MS2   AOD    D.Z2          ADVANCE BYTE ADDRESS        SCOP31A  00439
      1056   4002                LDI    D.Z2                                      SCOP31A  00440
      1057   1071                SHN    -6                                        MSG      00088
      1060   1760                SBN    60B     .CHAR. ILLEGAL IF GREATER THAN 60 MSG      00089
      1061   0617                PJN    MS4                                       SCOP31A  00441
      1062   4002                LDI    D.Z2          TEST SECOND CHARACTER OF BYTE MSG    00091
      1063   1277                LPN    77B                                       MSG      00092
      1064   1760                SBN    60B                                       MSG      00093
      1065   0613                PJN    MS4     .JUMP IF ILLEGAL                  MSG      00094
      1066   3703                SOD    D.Z3                                      SCOP31A  00442
      1067   0505                NJN    MS2           SENSE LESS THAN 5 BYTES PROCESSED SCOP31A 00443
      1070   3701                SOD    D.Z1                                      SCOP31A  00444
      1071   0415                ZJN    MS5           SENSE 40 BYTES PROCESSED    SCOP31A  00445
      1072   4002                LDI    D.Z2                                      SCOP31A  00446
      1073   0453                ZJN    MS3A          SENSE END OF MESSAGE-FILL WITH ZEROES SCOP31A 00447
      1074   3665                AOD    D.OUT+1                                   MSG      00102
```

```
1075   1063                SHN   -12                                                 MSG     00103
1076   3564                RAD   D.OUT             IF NOT,GET ADDRESS OF NEXT WORD   MSG     00104
1077                       UJK   MS1                                                 SCOP31A 00448
1100   2000 1142   MS4     LDC   MSG1        .MESSAGE FORMAT ERROR                   MSG     00113
1102   0200 0650           RJM   R.DFM        PROCESS DAYFILE MESSAGE                MSG     00114
1104   1413        MS12    LDN   M.ABORT                                             MSG     00115
1105   0322                UJN   MS7         .ABORT CP                               MSG     00116
1106   3052        MS5     LDD   D.PPIRB+2    IS THIS FOR DISPLAY ON SCOPE ONLY      MSG     00117
1107   0524                NJN   MS9         .YES, JUMP                              MSG     00118
1110   2000 1156           LDC   BUF                                                 SCOP31A 00449
1112   0200 0650           RJM   R.DFM             OUTPUT DAYFILE MESSAGE            SCOP31A 00450
1114   3062        MS10    LDD   D.IN                                                MSG     00123
1115   0411                ZJN   MS10A             NO RECALL                         MSG     00124
1116   3644                AOD   D.BA+4            SET COMPLETEION BIT FOR CPC       MSG     00125
1117   3053                LDD   D.PPIRB+3                                           MSG     00126
1120   1237                LPN   37B                                                 SCOP31A 00451
1121   1014                SHN   12                                                  SCOP31A 00452
1122   3154                ADD   D.PPIRB+4                                           MSG     00130
1123   0200 0634           RJM   R.TFL             RELOCATE ADDRESS                  SCOP31A 00453
1125   6240                CWD   D.BA              TELL CPC                          MSG     00131
1126   1412        MS10A   LDN   M.DPP                                               MSG     00132
1127   0200 0450   MS7     RJM   R.PROCES          SEND REQUEST TO OUTPUT REGISTER   MSG     00133
1131   0100 0100           LJM   R.IDLE            EXIT TO IDLE LOOP                 MSG     00134
1133   3051        MS9     LDD   D.PPIRB+1         EXTRACT CP NUMBER                 SCOP31A 00454
1134   1207                LPN   L.CPNUM                                             SCOP31A 00455
1135   1007                SHN   7                                                   SCOP31A 00456
1136   1630                ADN   W.CPDFM                                             MSG     00140
1137   6304 1156           CWM   BUF,D.Z4          WRITE MESSAGE FOR DISPLAY ONLY    SCOP31A 00457
1141   0352                UJN   MS10        .EXIT                                   MSG     00142
                   *                                                                 MSG     00143
1142   1505        MSG1    DIS   ,*MESSAGE FORMAT ERROR.*                            MSG     00144
1156   0000        BUF     DATA  0                                                   MSG     00145
1157                       END                                                       MSG     00146

030274 UNUSED STORAGE       139 STATEMENTS       475 SYMBOLS
```

11-2-4

BUF	0001156	001036, 001047, 001110, 001137
CH.DMP	0000023	
CH.FNT	0000015	
CH.FST	0000014	
CH.LIB	0000016	
CH.RBT	0000017	
CP.ECSM	0002034	
CP.IDLE	0002000	
CP.JOBN	0002020	
CP.MOVE	0002000	
CP.SM	0002023	
C.CPAR	0000001	
C.CPECFL	0000004	
C.CPEF	0000001	
C.CPERT	0000004	
C.CPFL	0000004	
C.CPFP	0000003	
C.CPFST	0000002	
C.CPNCSP	0000004	
C.CPNUM	0000001	
C.CPOUT	0000003	
C.CPPRI	0000000	
C.CPRA	0000003	
C.CPSM	0000002	
C.CPSTAT	0000000	
C.CPTIML	0000002	
C.DFBMS	0000001	
C.DIRCMA	0000001	
C.DIRFWA	0000000	
C.DIRPRU	0000004	
C.DIRPTR	0000000	
C.DIRRBA	0000002	
C.DIRRBN	0000003	
C.DIRUNT	0000001	
C.FCIB	0000000	
C.FCPNUM	0000003	
C.FCS	0000003	
C.FDC	0000002	
C.FEQP	0000000	
C.FFRBA	0000001	
C.FLBL	0000003	
C.FLOCK	0000003	
C.FLPRU	0000004	
C.FLRBEB	0000003	
C.FLRBWP	0000002	
C.FNAME	0000000	
C.FPDEV	0000002	
C.FPRI	0000004	
C.FSC	0000003	
C.FSDEV	0000001	
C.FTYPE	0000003	
C.PPFWA	0001000	000000
C.PPSWA	0002000	
C.PPTWA	0003000	
C.PP4WA	0004000	
C.PP5WA	0005000	
C.PP7WA	0007000	

SYMBOLIC REFERENCE TABLE

```
C.RBRA     0000003
C.RBRAD    0000000
C.RBRLAV   0000003
C.RBRTPA   0000000
C.RBRUNT   0000001
C.RBTAL    0000002
C.RBTFB    0000001
C.RBTLPR   0000000
C.RBTLRB   0000004
C.RBTPRU   0000003
C.RBTRBR   0000001
C.RBTWPL   0000000
C.RQSCS    0000000
C.RQSFS    0000002
C.RSTA     0000002
C.RSTRA    0000001
C.RSTU     0000004
C.RSTWP    0000003
C.RWPPCC   0000003
C.RWPPCF   0000000
C.RWPPLW   0000002
C.RWPPST   0000003
C.RWPPWC   0000004
C.RWPPWT   0000001
C.STAICP   0000002
C.STCPU    0000004
C.STEI     0000004
C.STFB     0000003
C.STU      0000003
C.STPFW    0000002
C.STPLW    0000004
C.STPMS    0000001
C.STPPRU   0000002
C.STPRBA   0000000
C.STPRBN   0000001
C.STPWC    0000000
D.BA       0000040          001015,  001022,  001025,  001116,  001125
D.CPAD     0000074
D.DTS      0000037
D.EST      0000032
D.FA       0000057
D.FF0      0000050
D.FF1      0000051
D.FF2      0000052
D.FF3      0000053
D.FF4      0000054
D.FF5      0000055
D.FF6      0000056
D.FF7      0000057
D.FIRST    0000060
D.FL       0000056
D.FNT      0000020
D.FR0      0000040
D.FR1      0000041
D.FR2      0000042
D.FR3      0000043
D.FR4      0000044
```

```
D.FR5      0000045
D.FR6      0000046
D.FR7      0000047
D.HN       0000071
D.IN       0000062            001020,  001114
D.JECS     0000045
D.JFL      0000037
D.JPR      0000046
D.JTL      0000047
D.LIMIT    0000066
D.OUT      0000064            001006,  001011,  001027,  001032,  001041,  001043,  001074,  001076
D.PPIR     0000075            001002
D.PPIRB    0000050            001003,  001004,  001010,  001016,  001106,  001117,  001122,  001133
D.PPMES1   0000077
D.PPONE    0000070            001001,  001047
D.PPOR     0000076
D.RA       0000055
D.SV0      0000070
D.SV1      0000071
D.SV2      0000072
D.SV3      0000073
D.SV4      0000074
D.SV5      0000075
D.SV6      0000076
D.SV7      0000077
D.SX0      0000060
D.SX1      0000061
D.SX2      0000062
D.SX3      0000063
D.SX4      0000064
D.SX5      0000065
D.SX6      0000066
D.SX7      0000067
D.TH       0000072
D.TH0      0000030
D.TH1      0000031
D.TH2      0000032
D.TH3      0000033
D.TH4      0000034
D.TH5      0000035
D.TH6      0000036
D.TH7      0000037
D.TR       0000073
D.TW0      0000020
D.TW1      0000021
D.TW2      0000022
D.TW3      0000023
D.TW4      0000024
D.TW5      0000025
D.TW6      0000026
D.TW7      0000027
D.T0       0000010
D.T1       0000011
D.T2       0000012
D.T3       0000013
D.T4       0000014
D.T5       0000015
```

14-34

SYMBOLIC REFERENCE TABLE

D.T6	0000016	
D.T7	0000017	
D.Z0	0000000	
D.Z1	0000001	001034, 001070
D.Z2	0000002	001040, 001055, 001056, 001062, 001072
D.Z3	0000003	001052, 001066
D.Z4	0000004	001035, 001137
D.Z5	0000005	
D.Z6	0000006	
D.Z7	0000007	
F.ERAR	0000002	
F.ERCP	0000004	
F.EROD	0000006	
F.ERPCE	0000005	
F.ERPP	0000003	
F.ERTL	0000001	
LE.FNT	0000003	
L.CPNUM	0000007	001134
L.PPHDR	0000005	
MSGA	0001033	001021
MSG1	0001142	001100
MS1	0001041	001077
MS10	0001114	001141
MS10A	0001126	001115
MS12	0001104	
MS2	0001055	001067
MS3	0001046	001014
MS3A	0001047	001053, 001073
MS4	0001100	001046, 001061, 001065
MS5	0001106	001071
MS7	0001127	001105
MS9	0001133	001107
M.ABORT	0000013	001104
M.AEWP	0000033	
M.CCPA	0000035	
M.CDF	0000011	
M.CPUST	0000036	
M.DCH	0000003	
M.DCP	0000016	
M.DEWP	0000023	
M.DFM	0000001	
M.DPP	0000012	001126
M.DTAPE	0000032	
M.EREQS	0000034	
M.NTIME	0000014	
M.OPDROP	0000030	
M.PAUSE	0000017	
M.PPTIME	0000004	
M.RCH	0000002	
M.RCLCP	0000021	
M.RCP	0000015	
M.REM	0000025	
M.REWP	0000022	
M.RPJ	0000037	
M.RPP	0000020	
M.RPRI	0000024	
M.RSTOR	0000010	

147-35

SYMBOLIC REFERENCE TABLE

M.RTAPE	0000031
M.SEF	0000030
M.STEP	0000005
N.CP	0000007
OV.ATS	0012423
OV.CHK	0031013
OV.CIO	0031117
OV.CKP	0031320
OV.CLL	0031414
OV.CLO	0031417
OV.CLS	0031423
OV.CPL	0032014
OV.CTS	0032423
OV.DIS	0041123
OV.DMP	0041520
OV.EXU	0053025
OV.LBC	0140203
OV.LDR	0140422
OV.LOC	0141703
OV.LOD	0141704
OV.MDI	0150411
OV.MEM	0150515
OV.MSG	0152307
OV.OPE	0172005
OV.PBC	0200203
OV.RBR	0220222
OV.REQ	0220521
OV.RFL	0220614
OV.RST	0222324
OV.SRB	0232202
OV.WBR	0270222
OV.1AJ	0340112
OV.1BJ	0340212
OV.1BT	0340224
OV.1CO	0340317
OV.1CY	0340331
OV.1DF	0340406
OV.1EE	0340505
OV.1EI	0340511
OV.1IX	0341130
OV.1LJ	0341412
OV.1LT	0341424
OV.1MR	0341522
OV.1MW	0341527
OV.1OD	0341704
OV.1OT	0341724
OV.1PO	0342017
OV.1RI	0342211
OV.1RO	0342217
OV.1RQ	0342221
OV.1SP	0342320
OV.1TD	0342404
OV.1XP	0343020
OV.2BP	0350220
OV.2BT	0350224
OV.2CA	0350301
OV.2CF	0350306

14-36

SYMBOLIC REFERENCE TABLE

OV.2CJ	0350312	
OV.2DF	0350406	
OV.2EF	0350506	
OV.2ES	0350523	
OV.2LA	0351401	
OV.2LB	0351402	
OV.2LE	0351405	
OV.2LP	0351420	
OV.2PC	0352003	
OV.2RC	0352203	
OV.2RQ	0352221	
OV.2RT	0352224	
OV.2TB	0352402	
OV.2TF	0352406	
OV.2TJ	0352412	
OV.2TR	0352422	
OV.2TS	0352423	
OV.2TW	0352427	
OV.2WT	0352724	
OV.30T	0361724	
OV.4LB	0371402	
OV.5DT	0400424	
OV.7TP	0422420	
O.BPRU	0000016	
O.RCHN	0000017	
O.RCMPR	0000002	
O.RDP	0000010	
O.RDPNP	0000011	
O.RDSK	0000001	
O.READ	0000000	
O.SKB	0000013	
O.SKF	0000012	
O.WRP	0000014	
O.WRPR	0000015	
O.WRT	0000004	
O.WRTR	0000005	
P.CAT	0000011	
P.CST2	0000015	
P.CST3	0000016	
P.CST4	0000017	
P.DFB	0000003	
P.EST	0000005	
P.FNT	0000004	
P.INS	0000007	
P.LIB	0000001	
P.RBR	0000002	
P.RBT	0000002	
P.RQS	0000013	
P.SPI	0000012	
P.ZERO	0000000	
R.CPFL	0000627	
R.CPRA	0000631	
R.DCH	0000714	
R.DFM	0000650	001102, 001112
R.EREQS	0000300	
R.ERQ	0000334	
R.IDLE	0000100	001131

(4-37)

SYMBOLIC REFERENCE TABLE

R.MTR	0000450	
R.OVL	0000124	
R.OVLJ	0000111	
R.PAUSE	0000430	
R.PROCES	0000450	001127
R.RCH	0000704	
R.READP	0000460	
R.RWP	0000505	
R.RWPP	0000530	
R.STB	0000620	
R.STBMSK	0000611	
R.STEP	0000734	
R.TFL	0000634	001012, 001044, 001123
R.WAIT	0000410	
R.WRITEP	0000470	
S.DIRPR	0000010	
S.DIRPT	0000004	
S.FNTEQP	0000006	
S.FNTLK	0000005	
S.FNTTYP	0000003	
S.RBRUNT	0000006	
S.RBTRBR	0000003	
S.RBTREL	0000007	
S.RBTRNU	0000006	
S.STF	0000006	
S.STFA	0000000	
S.STFEOF	0000004	
S.STFETP	0000002	
S.STFNTP	0000003	
S.STFPRI	0000005	
S.STFREL	0000004	
T.CIDLE	0000023	
T.CLK	0000030	
T.CPA1	0000200	
T.CPA2	0000400	
T.CPA3	0000600	
T.CPA4	0001000	
T.CPA5	0001200	
T.CPA6	0001400	
T.CPA7	0001600	
T.CPS	0000040	
T.CPT1	0000056	
T.CPZ	0000020	
T.CP8	0002000	
T.CST	0000014	
T.DATE	0000031	
T.JDATE	0000027	
T.MON	0000021	
T.MSC	0000040	
T.MSP	0000037	
T.PIDLE	0000024	
T.PPC1	0000060	
T.PPC2	0000070	
T.PPC3	0000100	
T.PPC4	0000110	
T.PPC5	0000120	
T.PPC6	0000130	

8-3-71

SYMBOLIC REFERENCE TABLE

T.PPC7	0000140	
T.PPC8	0000150	
T.PPC9	0000160	
T.PPR	0000025	
T.PPS0	0000041	
T.PPS1	0000042	
T.PPS2	0000043	
T.PPS3	0000044	
T.PPS4	0000045	
T.PPS5	0000046	
T.PPS6	0000047	
T.PPS7	0000050	
T.PPS8	0000051	
T.PPS9	0000052	
T.SLAB1	0000031	
T.SLAB2	0000032	
T.SLAB3	0000033	
T.SLAB4	0000034	
T.SLAB5	0000035	
T.SLAB6	0000036	
T.STATCP	0000056	
T.STO	0000022	
T.TMP	0000055	
T.UAS	0000056	
W.CPAR	0000157	
W.CPCAF	0000051	
W.CPCAL	0000150	
W.CPDFM	0000030	001136
W.CPECS	0000022	
W.CPEF	0000020	
W.CPENC	0000155	
W.CPERT	0000040	
W.CPFL	0000020	
W.CPJNAM	0000021	
W.CPOAE	0000153	
W.CPOUT	0000153	
W.CPPRI	0000022	
W.CPRCL	0000025	
W.CPRES1	0000156	
W.CPRES2	0000160	
W.CPSM	0000020	
W.CPSTAT	0000020	
W.CPTBUF	0000041	
W.CPTBUL	0000050	
W.CPTIME	0000023	
W.CPTIML	0000022	
W.CPVRNO	0000154	
W.EQP	0000027	
W.FSTCC	0000151	
W.FSTNR	0000152	
W.FTYPE	0000000	
W.PPIR	0000000	
W.PPMES1	0000002	
W.PPMES2	0000003	
W.PPMES3	0000004	
W.PPMES4	0000005	
W.PPMES5	0000006	

14-37

SYMBOLIC REFERENCE TABLE

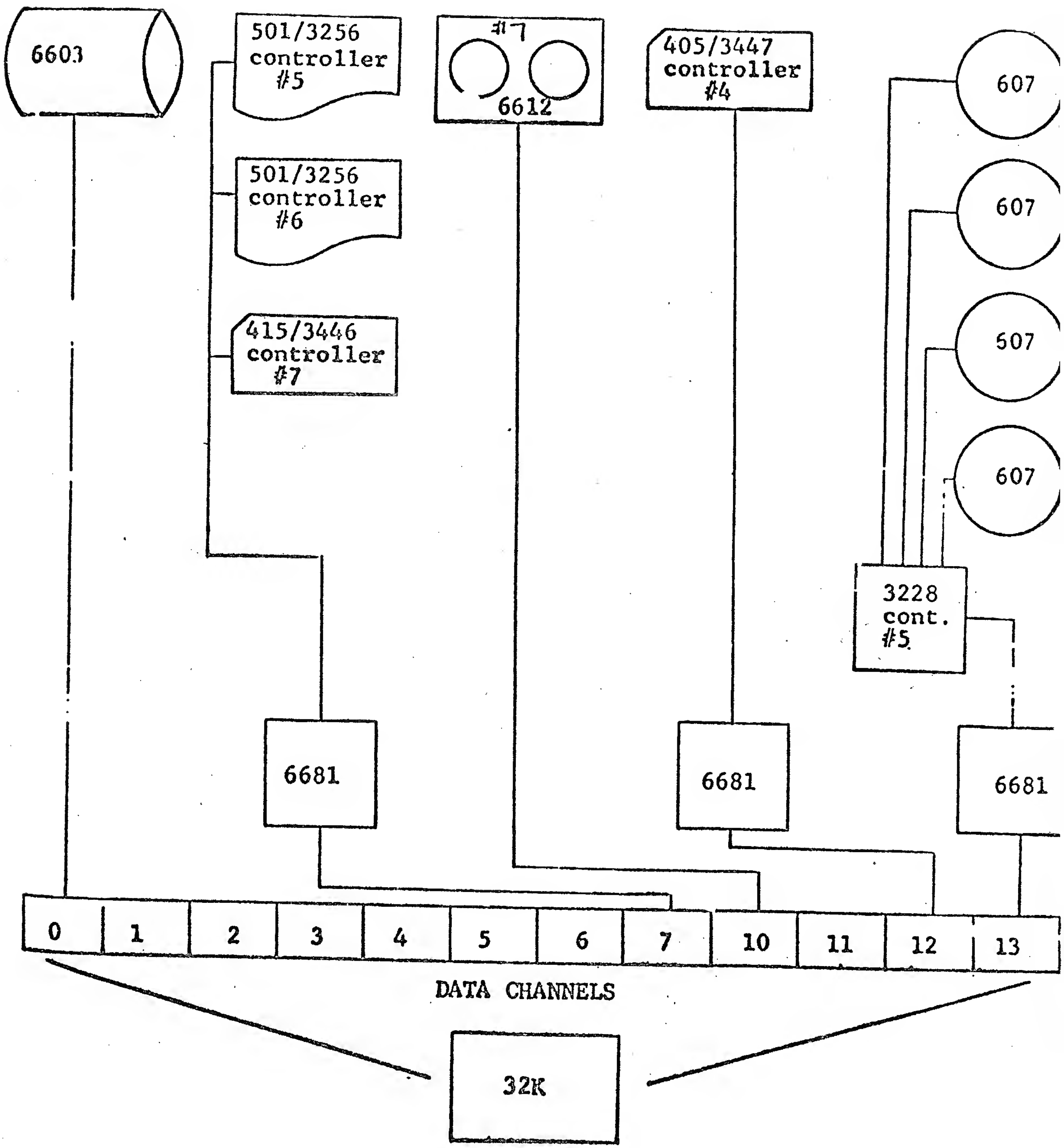
W.PPMES6	0000007
W.PPOR	0000001
W.PPTIME	0000024
W.RBRLAV	0000001
W.RBRTPA	0000000
W.RBRUNT	0000000
W.RWPPCW	0000002
W.SSW	0000026
W.STCPU	0000000
W.STEI	0000000
W.STFB	0000001
W.STO	0000000
W.STPFW	0000001
W.STPLW	0000001
W.STPMS	0000001
W.STPPRU	0000000
W.STPRBA	0000000
W.STPRBN	0000000
W.STPWC	0000001

1/4-40

```
02/08/68  SCOPE V3.1 6500-00 131K 11/21/67
01.25.36.SCPDOIF
01.25.36.SCPDOC,P17,T500,CM60000.
01.25.52.REQUEST(OLDPL)
01.25.52. (53 ASSIGNED)
01.25.52.REWIND(OLDPL)
01.25.53.UPDATE(Q)
01.26.13. UPDATING FINISHED
01.26.13.COMPASS(I=COMPILE,S=SCPTEXT)
01.26.23.CP  013.068 SEC.
01.26.23.PP  041.638 SEC.
```

UNIVERSITY OF ARIZONA

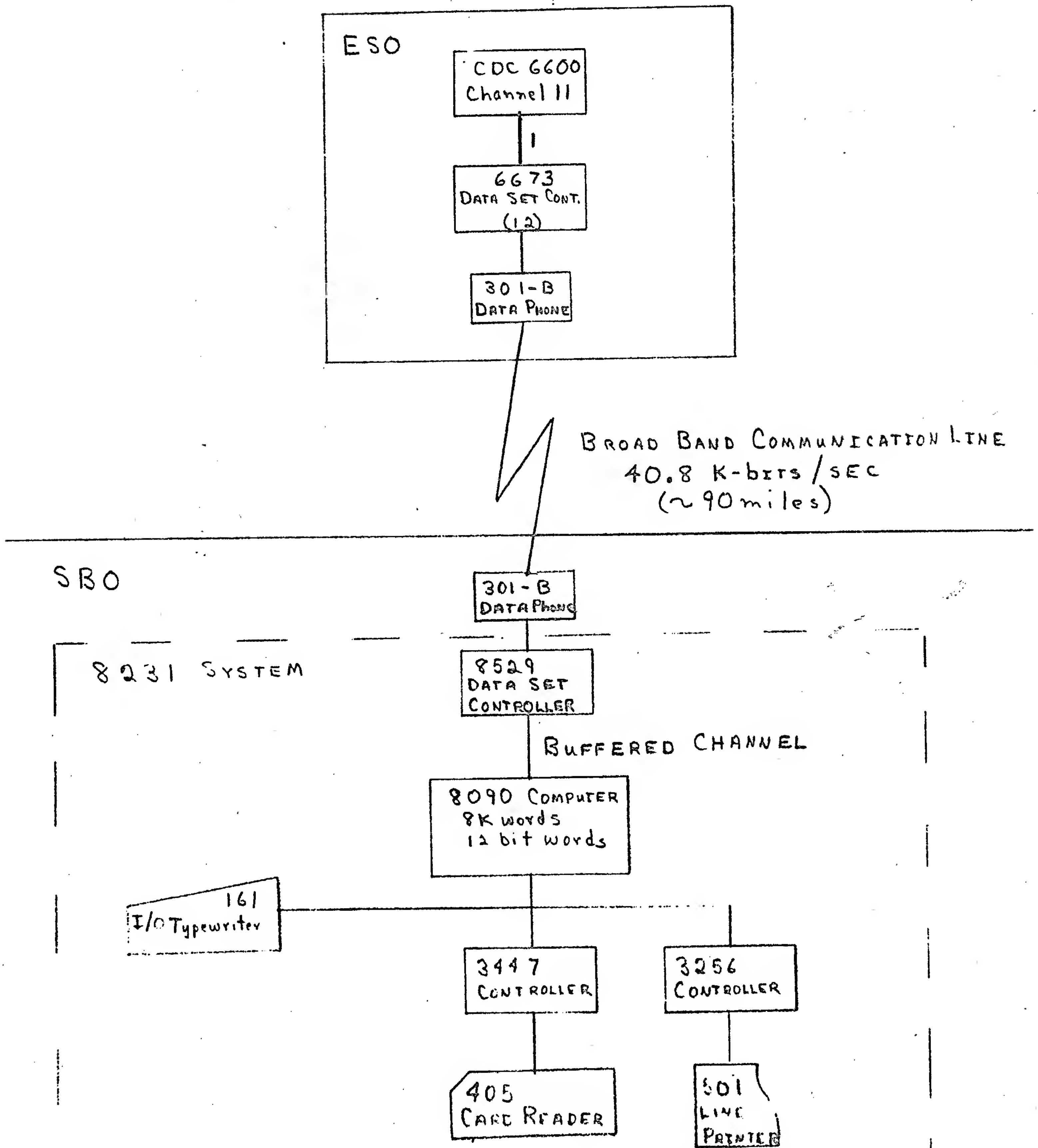
6400



AEROSPACE CONFIGURATION

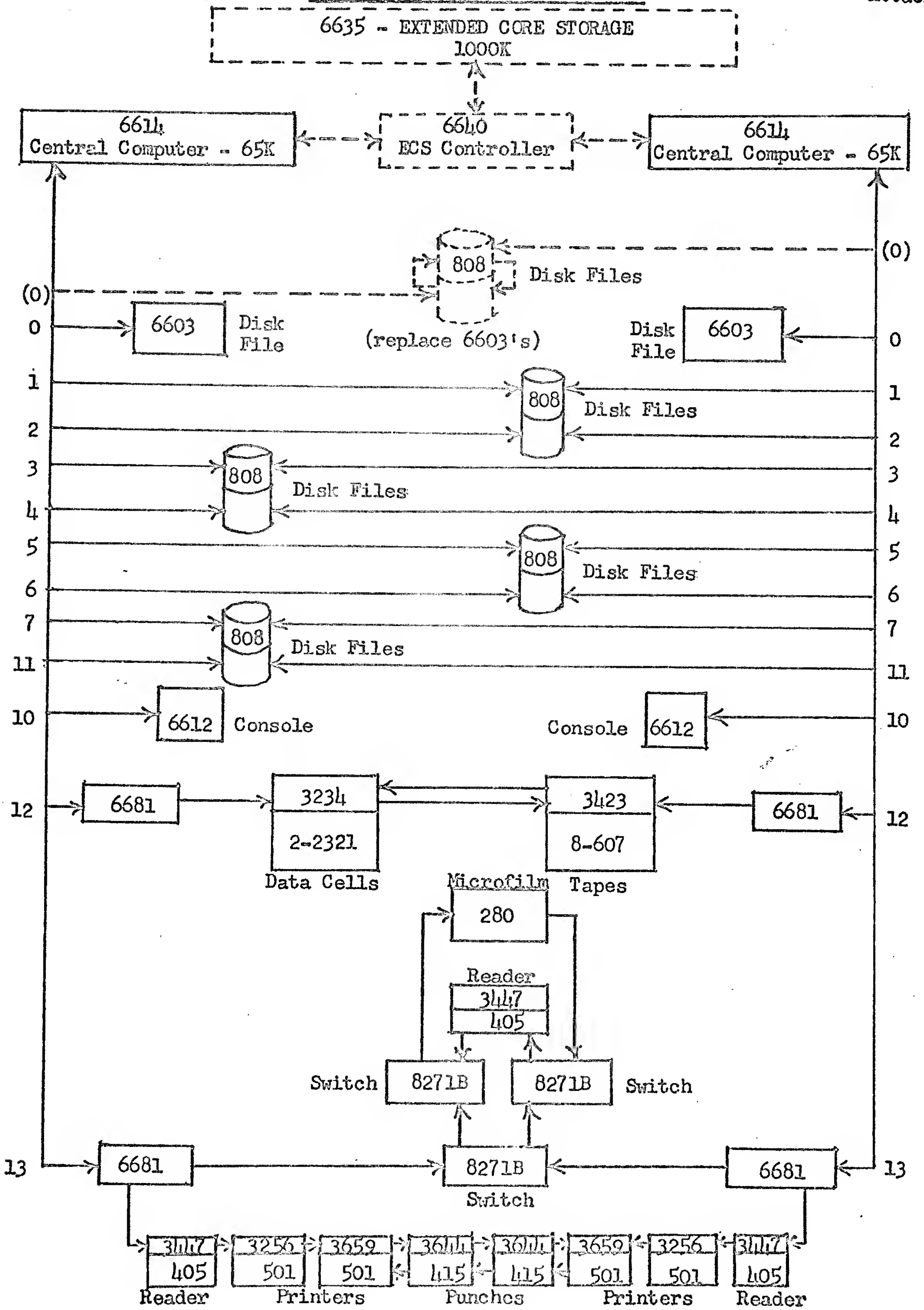
SBO REMOTE

3/1/68
RKV
2 of 2



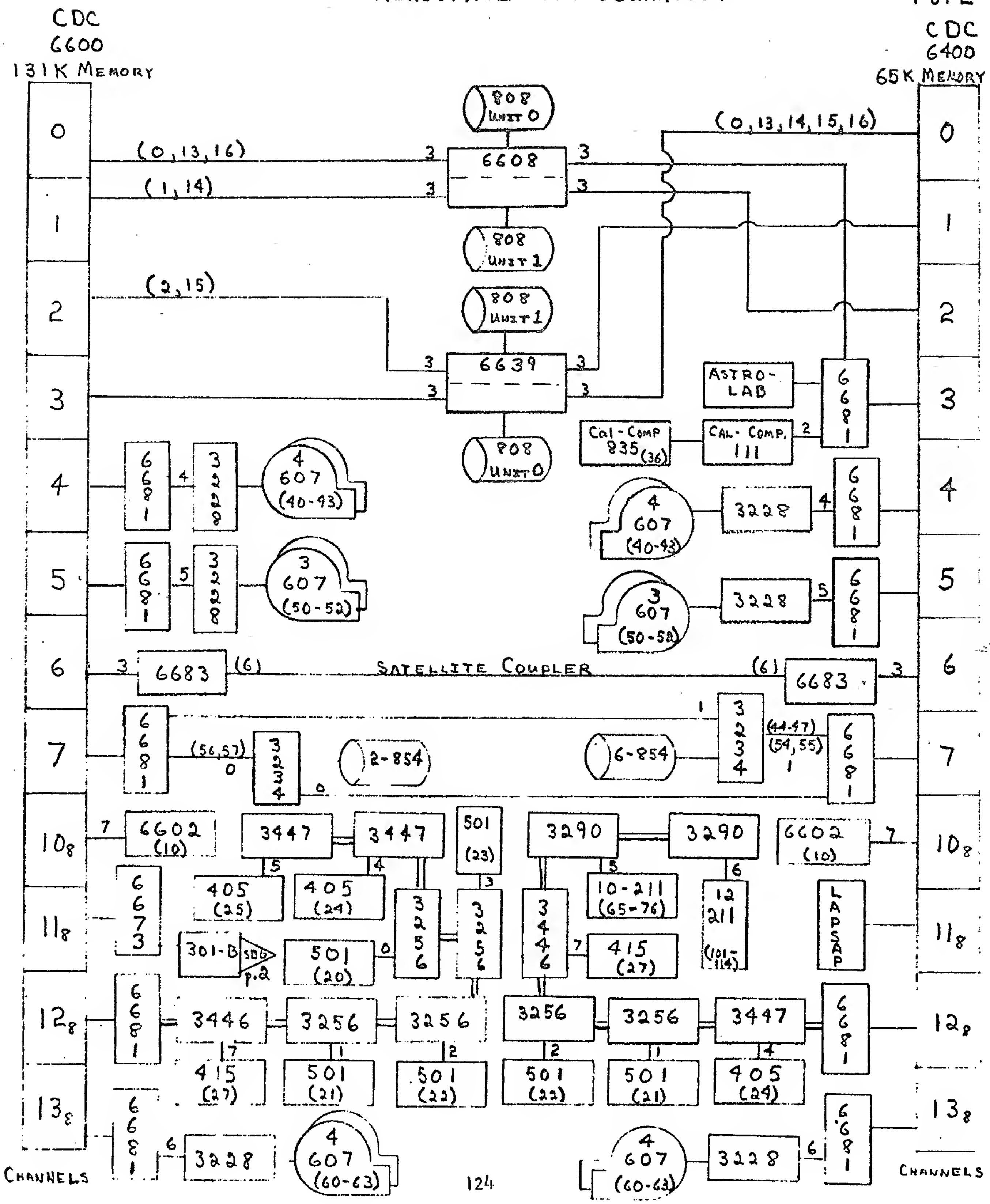
BETTIS ATOMIC POWER LABORATORY

Attachment

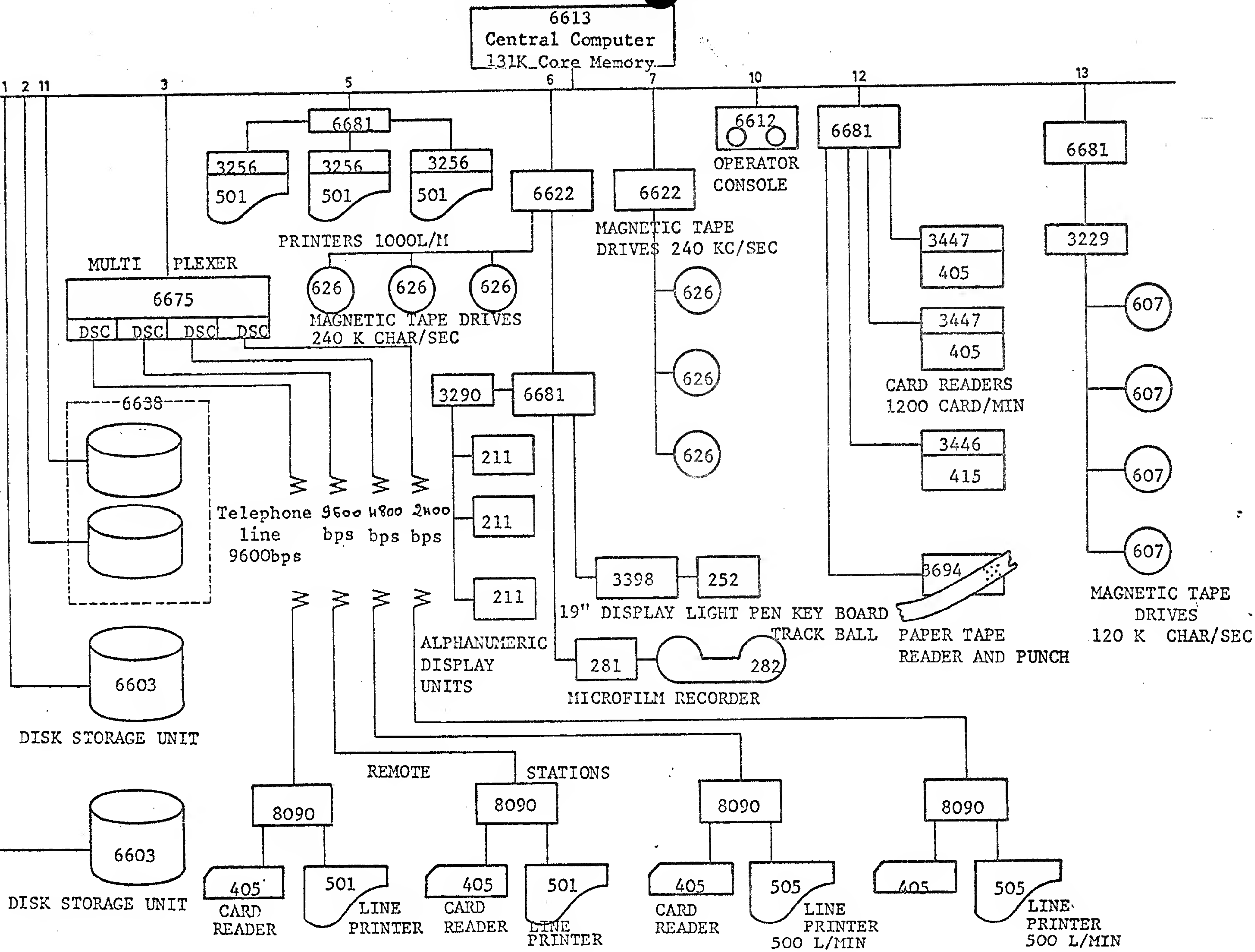


3/1/68
RKV
1 of 2

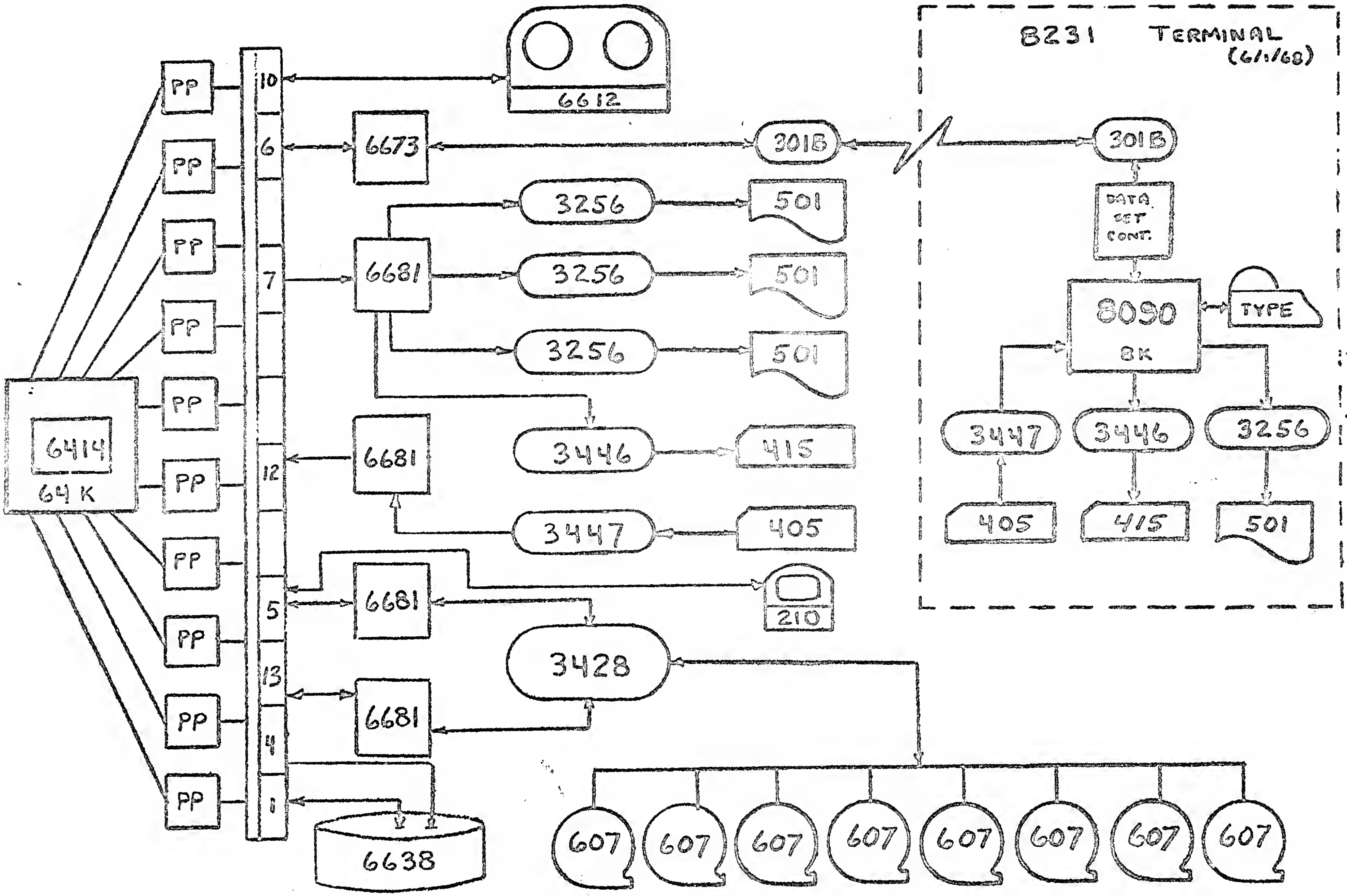
AEROSPACE CONFIGURATION



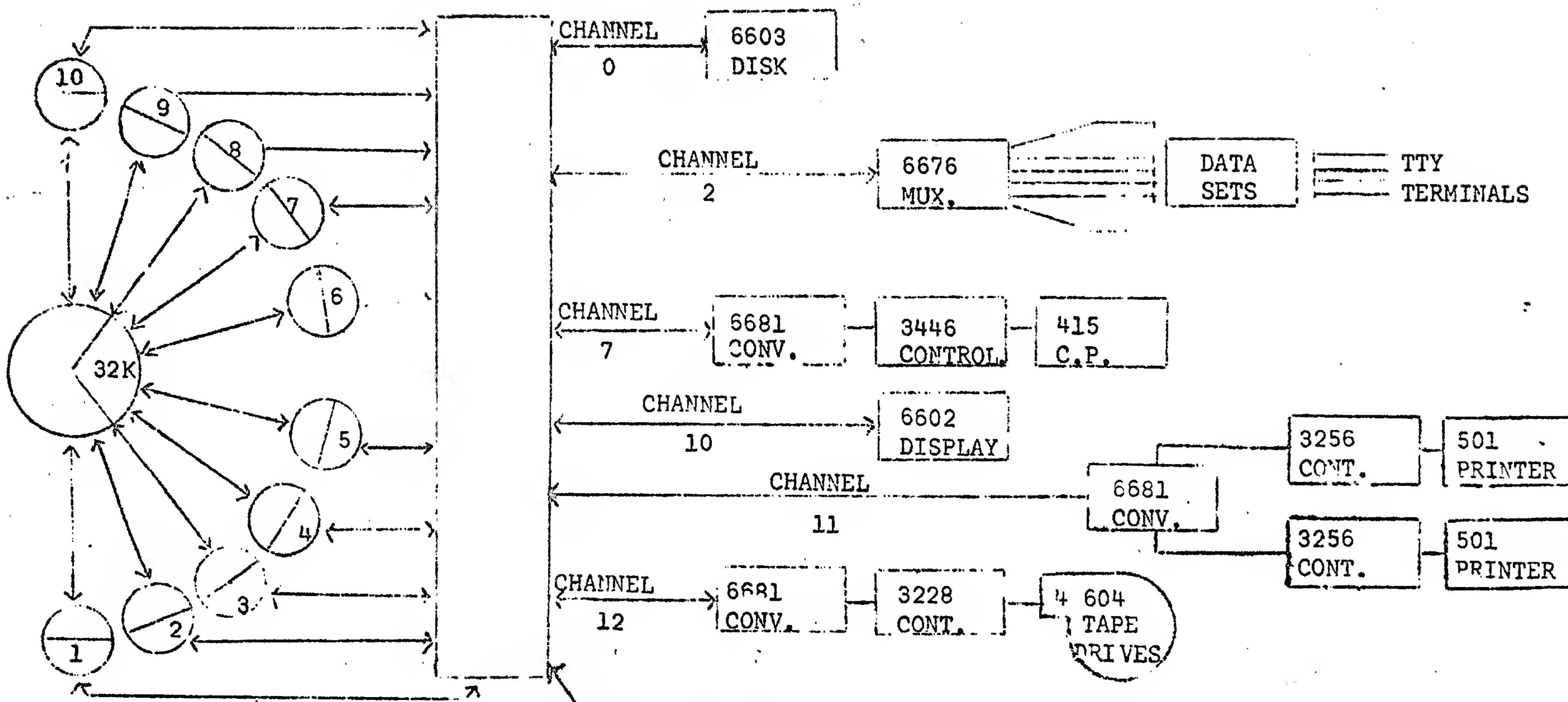
ELECTRICITE DE FRANCE



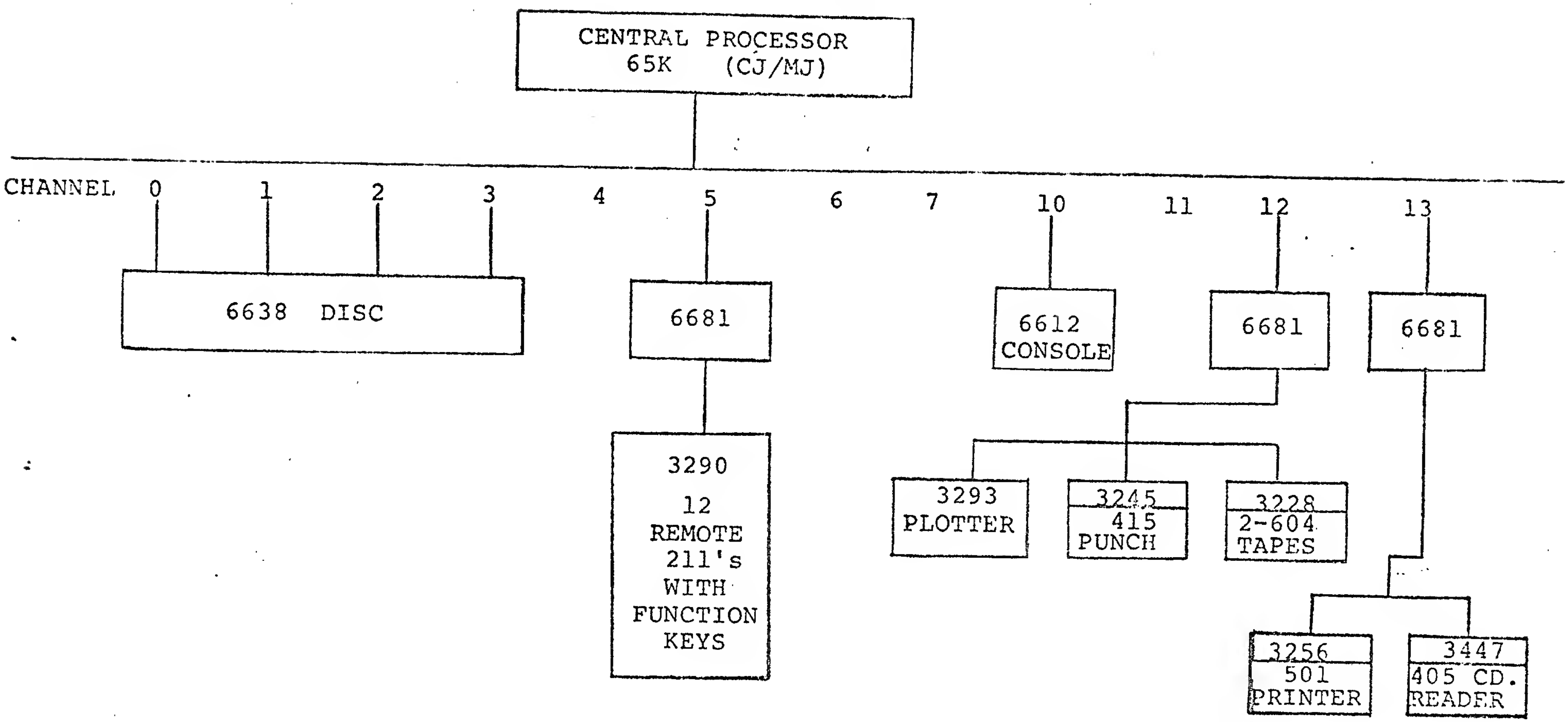
CDC 6400 Scientific Computer System

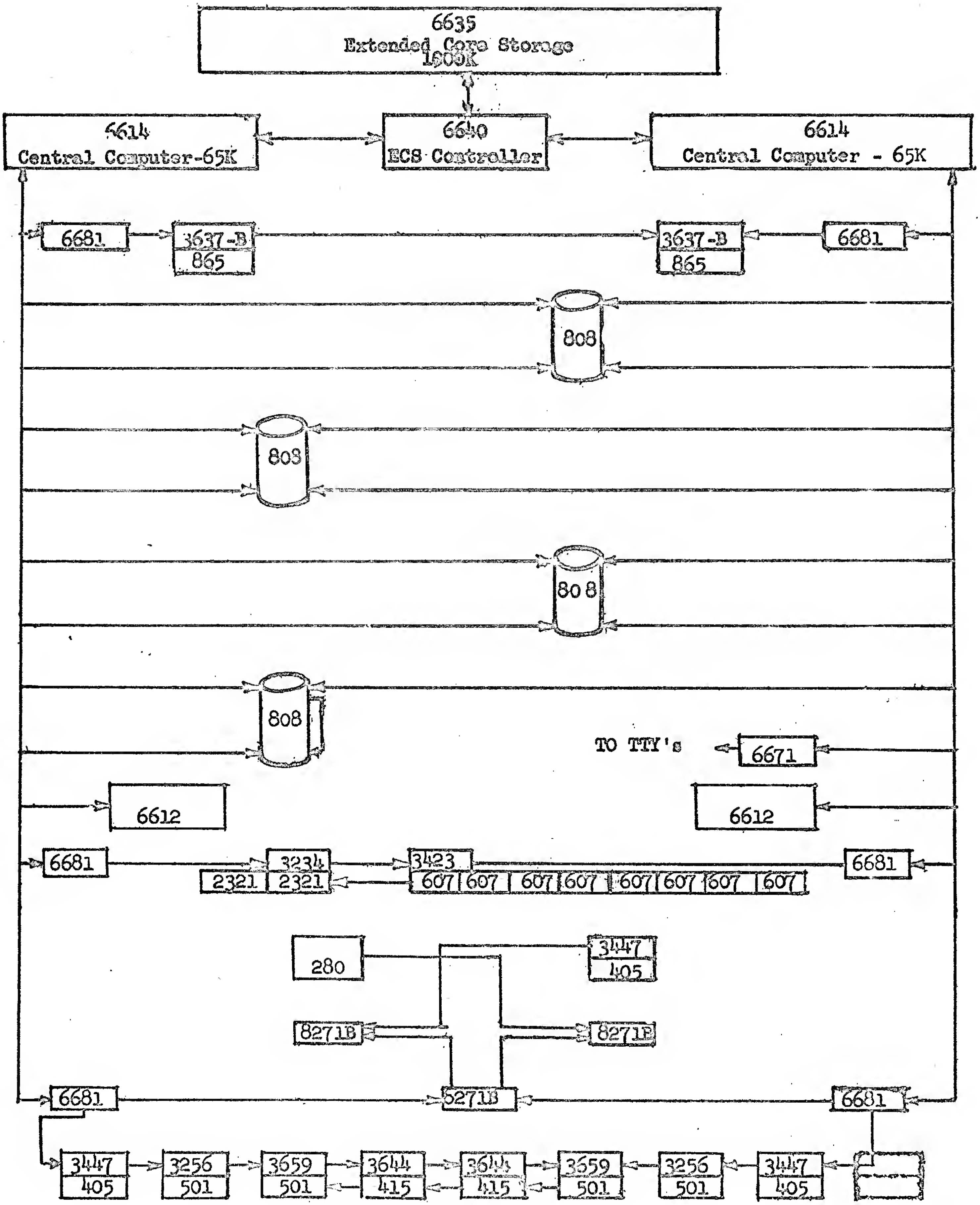


FSU CONFIGURATION

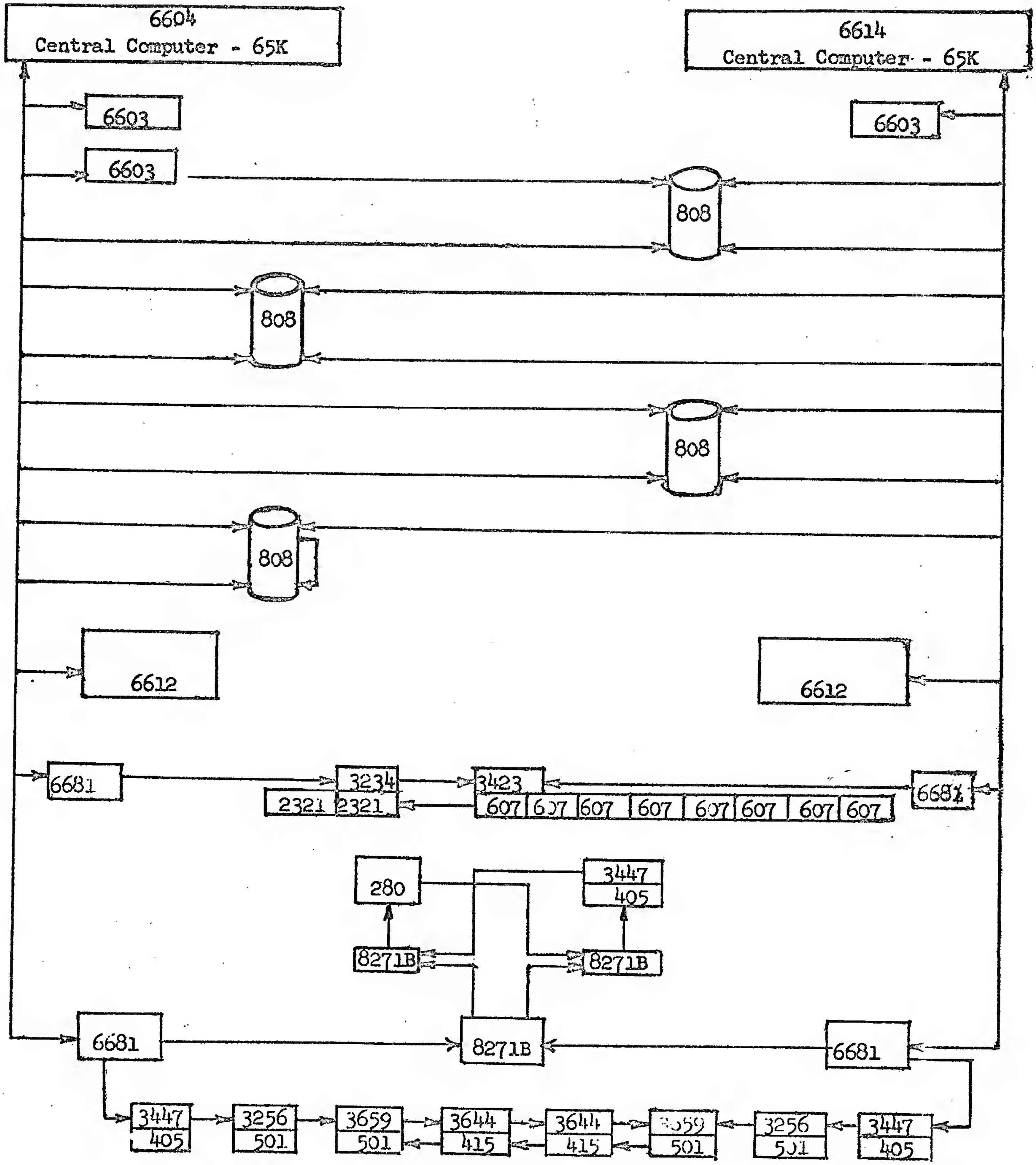


IDA COMPUTER FACILITY



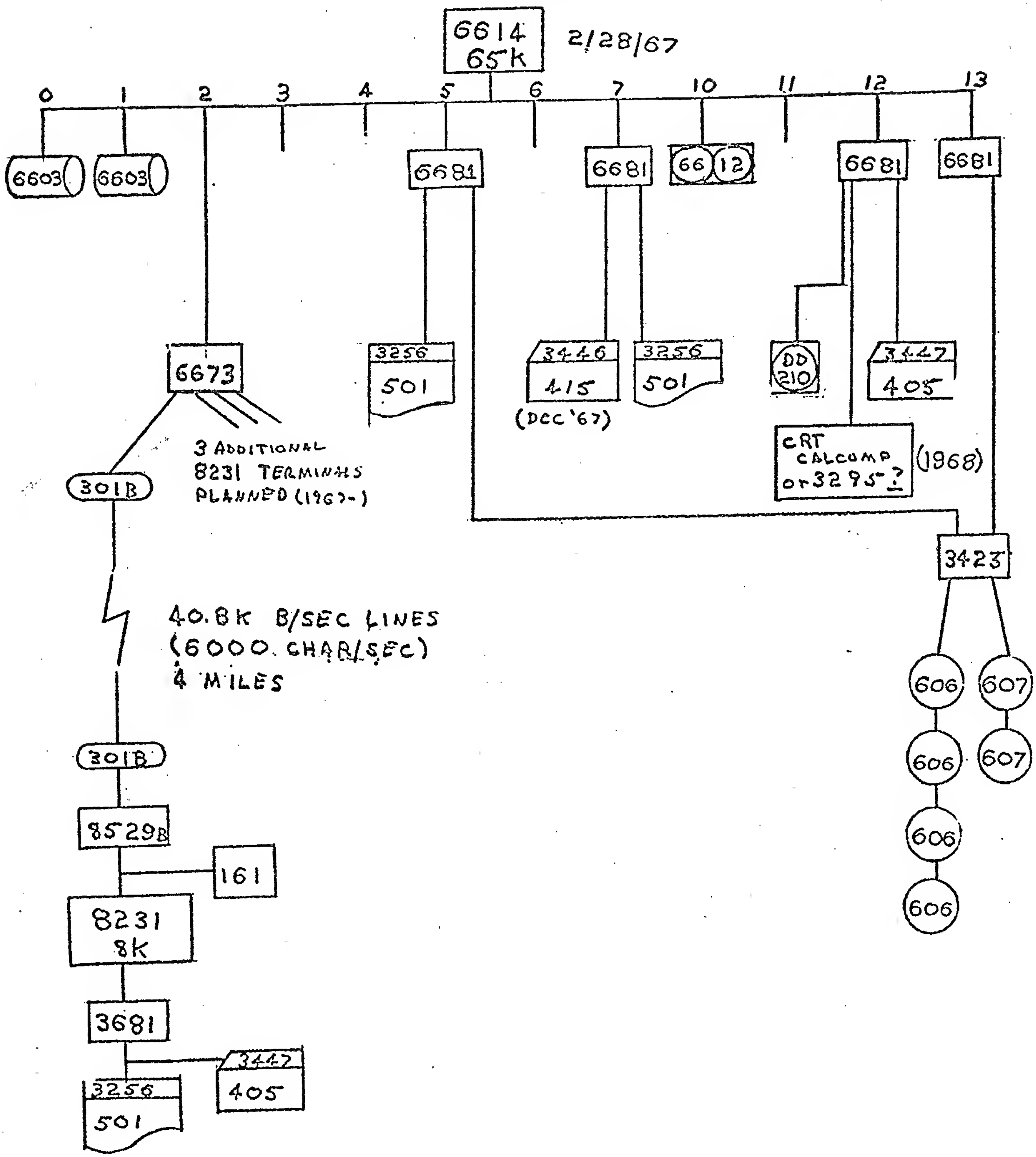


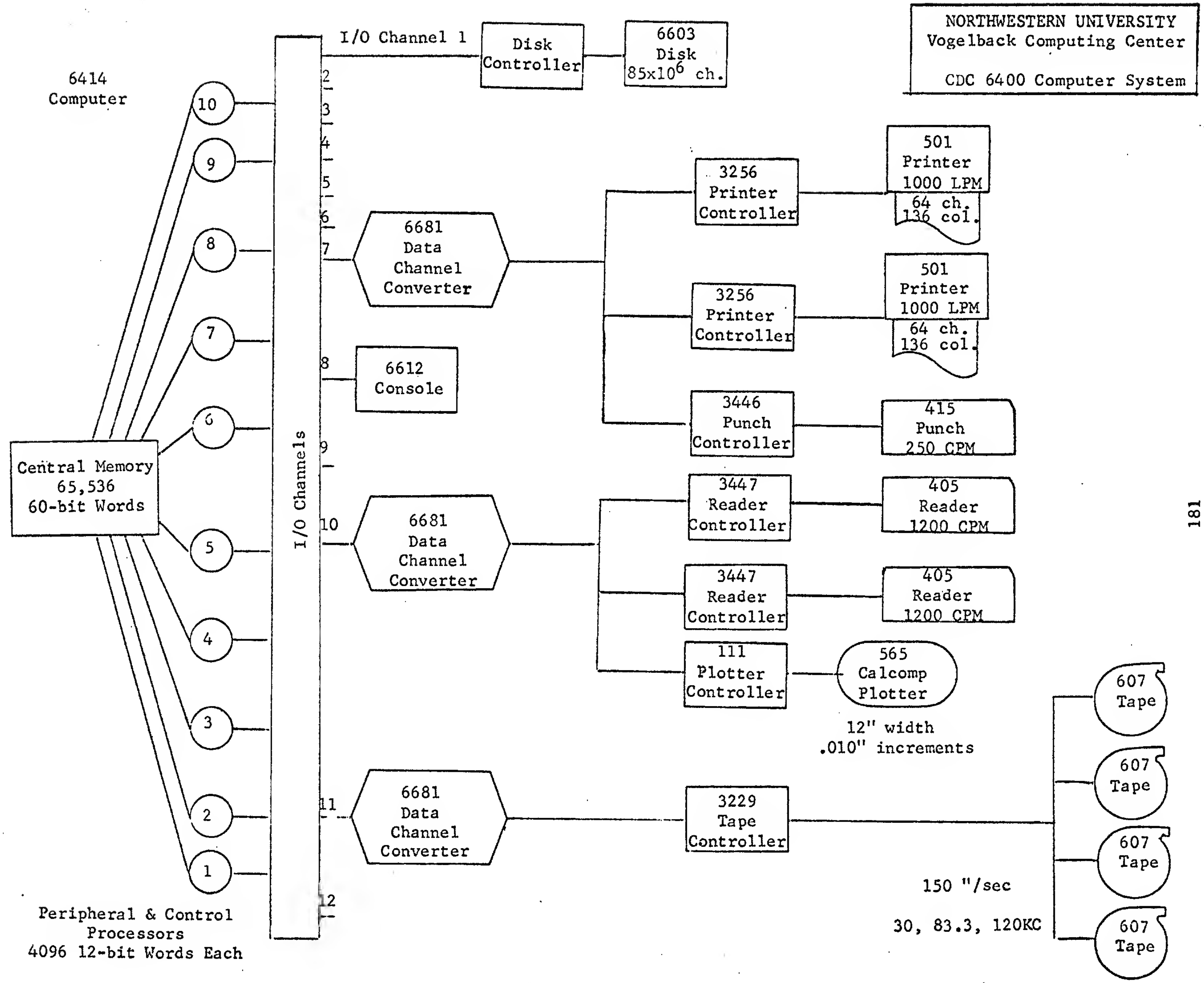
KNOLLS ATOMIC POWER LABORATORY PLANNED CONFIGURATION



KNOLLS ATOMIC POWER LABORATORY CURRENT CONFIGURATION

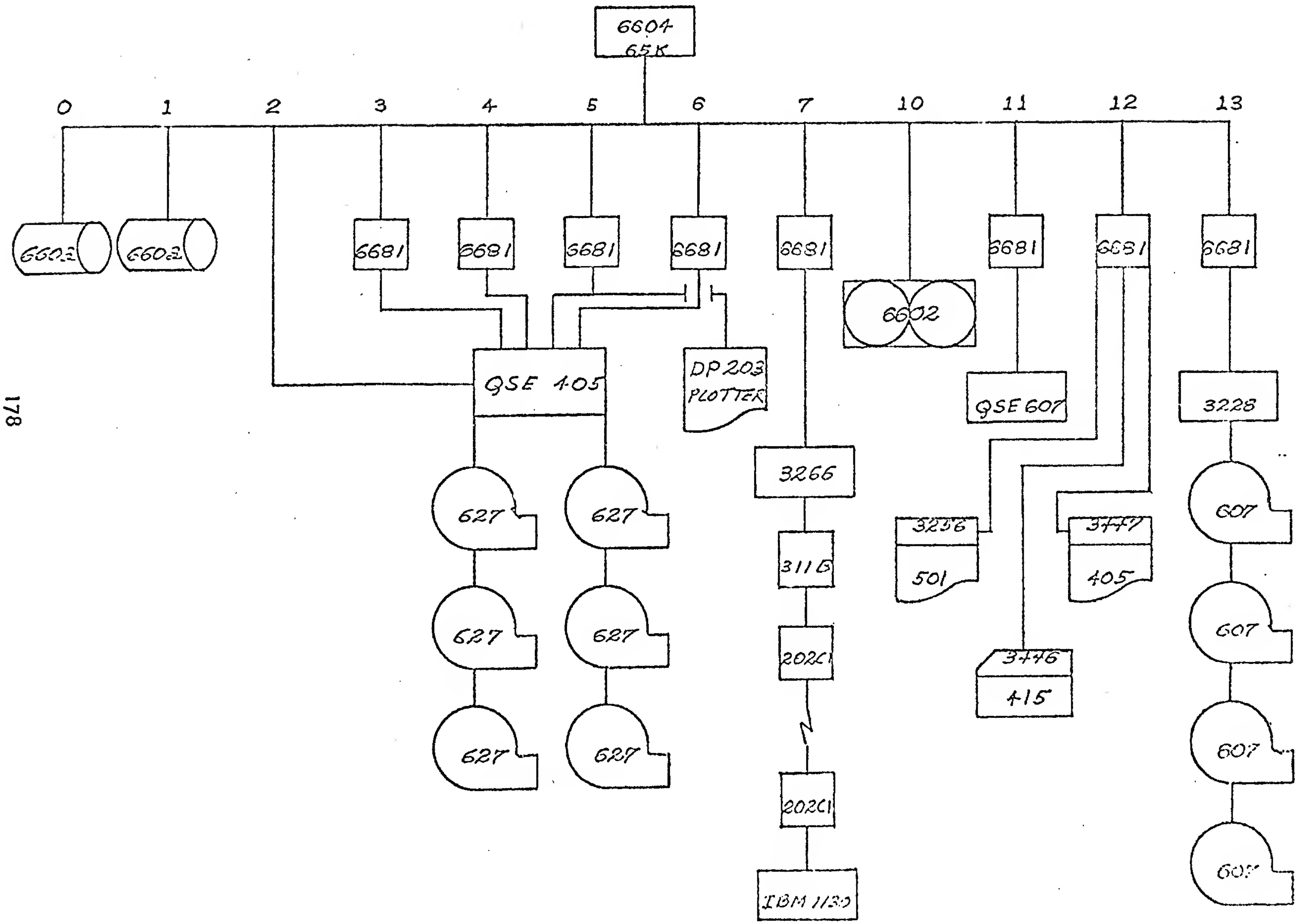
UNIVERSITY OF MINNESOTA
MINNEAPOLIS, MINNESOTA



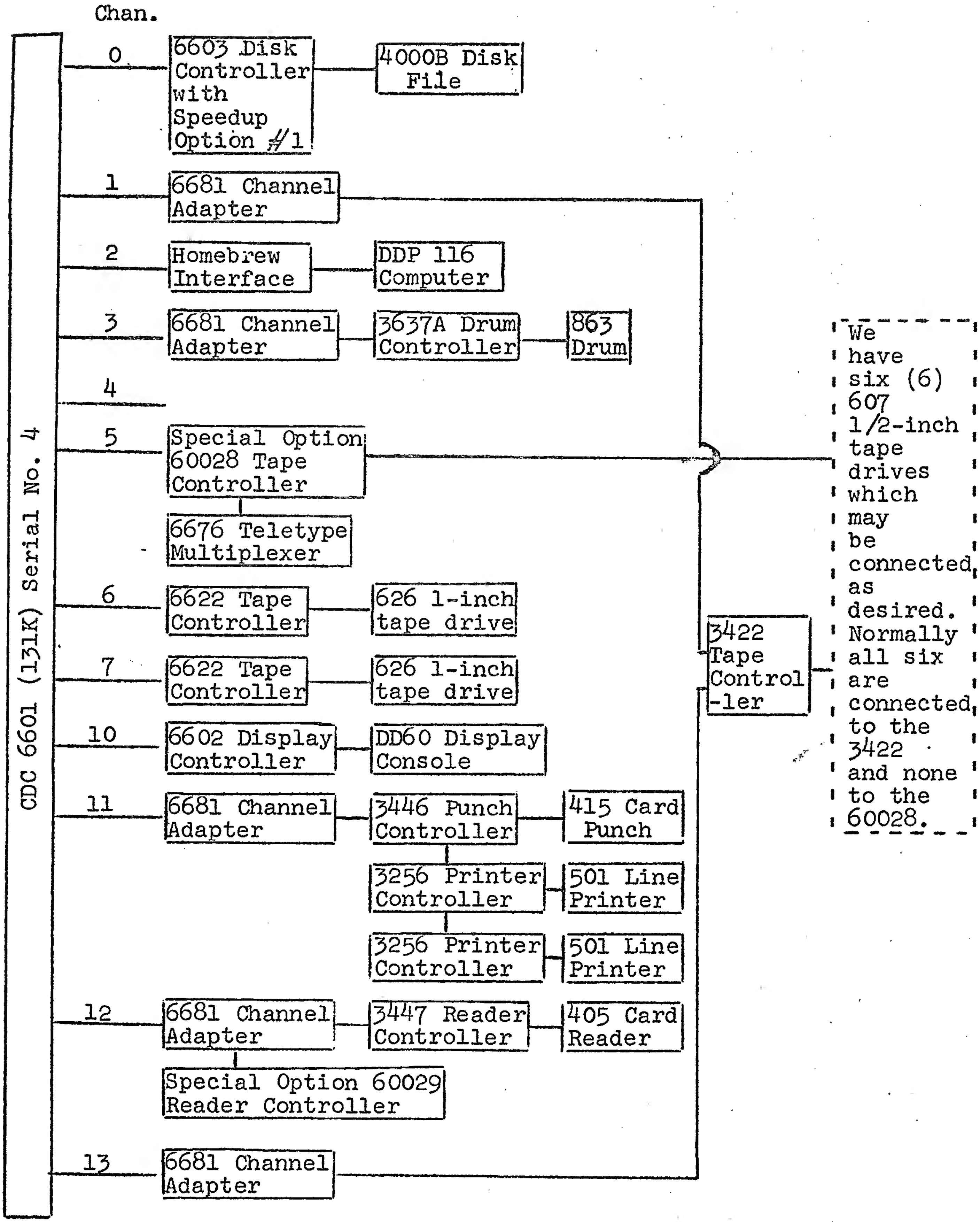

NORTHWESTERN UNIVERSITY
Vogelback Computing Center
CDC 6400 Computer System
6414 Computer
Central Memory 65,536 60-bit Words
10
9
8
7
6
5
4
3
2
1
Peripheral & Control Processors 4096 12-bit Words Each
I/O Channels
I/O Channel 1
2
3
4
5
6
7
8
9
10
11
12
Disk Controller
6603 Disk 85×10^6 ch.
6681 Data Channel Converter
6612 Console
6681 Data Channel Converter
6681 Data Channel Converter
3256 Printer Controller
3256 Printer Controller
3446 Punch Controller
3447 Reader Controller
3447 Reader Controller
111 Plotter Controller
3229 Tape Controller
501 Printer 1000 LPM
64 ch. 136 col.
501 Printer 1000 LPM
64 ch. 136 col.
415 Punch 250 CPM
405 Reader 1200 CPM
405 Reader 1200 CPM
565 Calcomp Plotter
12" width .010" increments
607 Tape
607 Tape
607 Tape
607 Tape
150 "/sec
30, 83.3, 120KC

MOBIL OIL CORPORATION (MOGS)

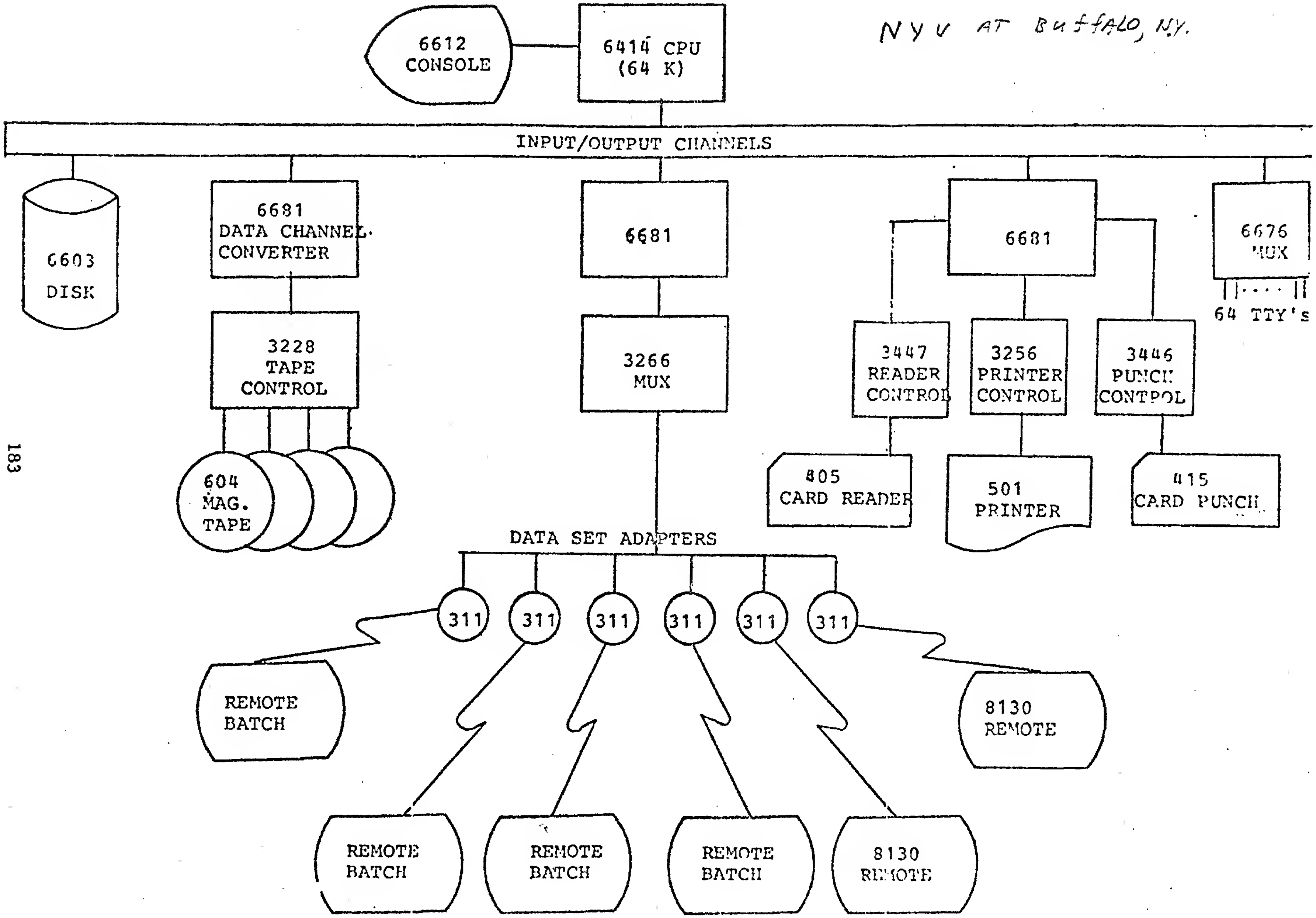
Geophysical Services Center
1930 Proctor Street
Dallas, Texas



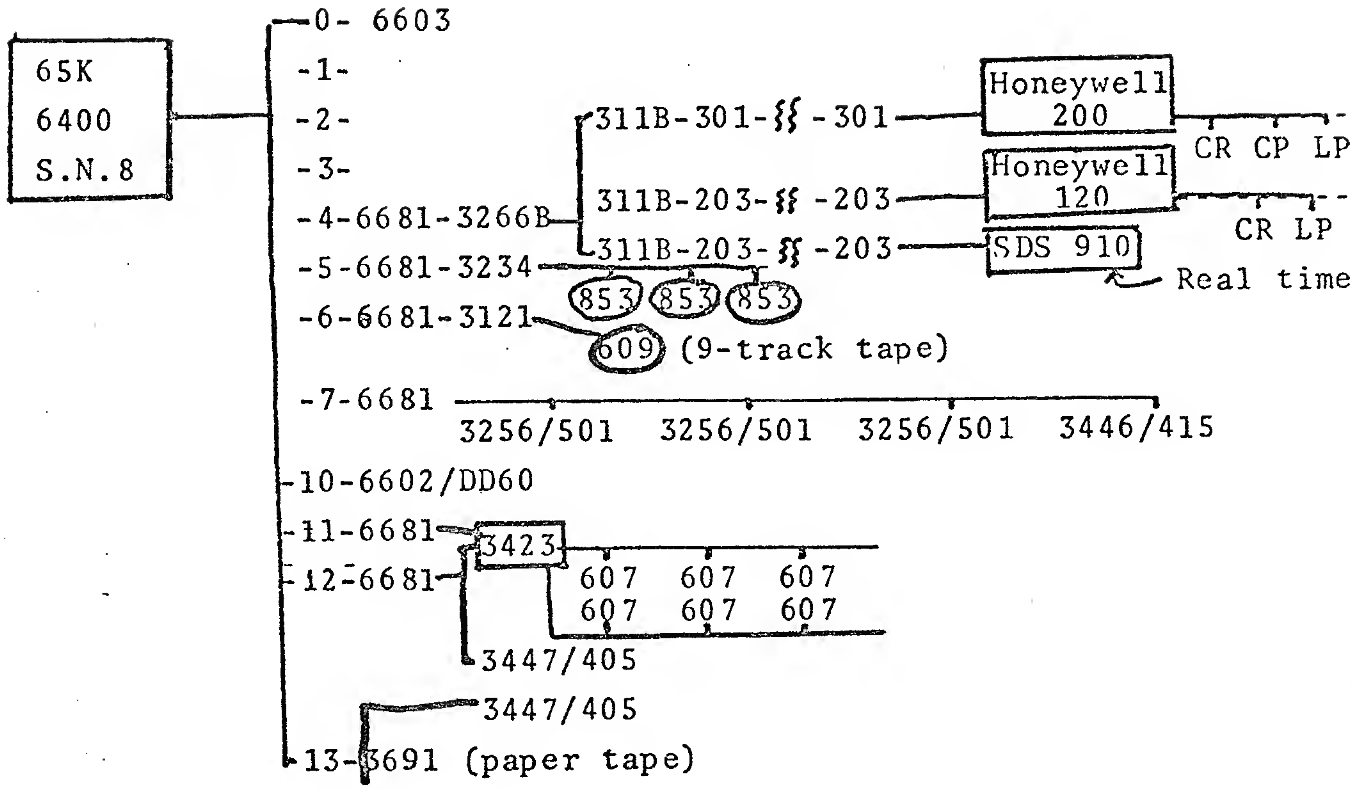
NYU -- Courant Institute CDC 6600 Channel Configuration (3/5/68)



NYU AT Buffalo, N.Y.



SAO CONFIGURATION



WESTINGHOUSE TELE-COMPUTER CENTER

CDC INSTALLATION